

For Reference

NOT TO BE TAKEN FROM THIS ROOM

Ex LIBRIS
UNIVERSITATIS
ALBERTAENSIS



THE UNIVERSITY OF ALBERTA

INDUCTIVE RESOLUTION

BY



CHARLES GRADY MORGAN

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

SPRING, 1972



Digitized by the Internet Archive
in 2019 with funding from
University of Alberta Libraries

<https://archive.org/details/Morgan1972>

Thesis
1972
103

THE UNIVERSITY OF ALBERTA
FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled Inductive Resolution submitted by Charles Grady Morgan in partial fulfilment of the requirements for the degree of Master of Science.

THE UNIVERSITY OF CHICAGO

DEPARTMENT OF CHEMISTRY

THE UNIVERSITY OF CHICAGO
DEPARTMENT OF CHEMISTRY
5408 S. DICKINSON DRIVE
CHICAGO, ILLINOIS 60637
(773) 936-3333

Abstract

The importance of inductive procedures in artificial intelligence is discussed in order to give the motivation for the thesis. Two main points are made: first, inductive procedures play a major role in the intelligent behavior of humans and thus the development of intelligent machines seems to require the implementation of inductive procedures; second, inductive reasoning frequently plays a central role in human theorem proving, and the implementation of the "reason backwards" heuristic requires the implementation of inductive procedures. The syntax and semantics of first order predicate calculus are briefly sketched; the thesis deals exclusively with the language of first order predicate calculus.

Standard systems of deductive inferences, which lead from axioms to consequences, are truth preserving systems. The importance of falsehood preserving inferences is established by noting that such inferences lead from consequences to axioms for those consequences. That is, falsehood preserving inferences are inductive. A general method for converting standard truth preserving systems into falsehood preserving systems is established. It is proved that the method preserves the properties of completeness and consistency of the original system.

The general conversion method is then applied to the Resolution Principle. An inductive Resolution Principle is developed and it is proved to have meta-properties analogous to those of the deductive

Resolution Principle.

A new proof procedure which combines deduction and induction is then discussed. It is argued that for any specific problem, the new procedure is at least as fast (in terms of number steps in the proof) as either induction or deduction. Some problems for further research on these topics are also outlined.

Acknowledgment

I would like to express my sincere appreciation to the members of the examining committee without whose helpful advice this thesis would be even more unreadable than it is. Professor Chen kindly agreed to supervise the research and waited most patiently for the results. Professor Sampson's expressions of confidence and suggestions for revisions have been extremely helpful. Professor Rozeboom did not allow friendship to dull his critical eye and thus saved me from making several mistakes. One could not have asked for a better committee.

Table of Contents

	Page
Chapter I: Introduction	1
A. Induction and Deduction	1
B. The Formal Language	7
Chapter II: F-Rules	14
A. Preliminary Comments and Assumptions	14
B. Some Difficulties with Meltzer's Procedure	15
C. Induction and Falsehood Preserving Rules	18
D. Obtaining Falsehood Preserving Rules	21
E. Advantages	32
Chapter III: Resolution	34
A. Deductive Resolution	34
B. Inductive Resolution	50
Chapter IV: Power	62
A. The Parallel Proof Procedure	62
B. Parallel Proof and Resolution	67
C. Problems for Further Research	70
Bibliography	73
Appendix	76

Chapter I: Introduction

A. Induction and Deduction

Attempting to simulate human intellectual activity by heuristic programming techniques has proven to be one of the most fruitful approaches in the field of artificial intelligence. The first heuristic programme to be fully realized on a computer was the Logic Theorist (sometimes referred to as the Logic Theory Machine), which was implemented by Newel, Shaw, and Simon in 1956 (see [11]). The problem with which the Logic Theorist was concerned was that of finding proofs of theorems in sentential calculus given the axioms, rules of inference, and the proposed theorem. The Logic Theorist dealt with the version of sentential calculus presented in Principia Mathematica and was tested by having it attempt to find proofs for the major theorems found in the second chapter of that work (see [18]).

Rather than striving for efficiency, the authors of the Logic Theorist were attempting to mirror the processes which a human logician would employ in proving theorems. The Logic Theorist was not designed as a practical tool to be employed in doing deductions, solving problems, or finding new theorems. The authors instead hoped to illustrate the procedures employed by humans.

McCarthy has stated that the ability to make some deductive inferences from information supplied is a necessary (though not sufficient) component of any device that can be said to have "common sense".

In addition to being interesting for its own sake, the ability to make such inferences is very useful for other purposes in artificial intelligence. For example, most efficient question answering programmes depend heavily on automatic theorem proving capabilities (for example, see Chapter 10 of [16]). General problem solving procedures also frequently employ theorem proving components (for example, see Chapter 7 of [12]). Because of the wide applicability of automatic theorem proving to other problems and because of its great potential usefulness as a tool in mathematics and other formalistic areas, in recent years more emphasis has been placed on the ability to efficiently deduce consequences. That is, most researchers are no longer concerned with mirroring human thought processes in theorem proving. Rather, they are more concerned with developing programmes which are quick and efficient.

There are two separate aspects of theorem proving which we will find it useful to distinguish. First, we may be given a set of axioms and an additional statement S , and our problem is that of finding a proof of S from the axioms or other given statements. We may or may not be given the information that such a proof exists (i.e., that S is a theorem). In either case, we would simply carry out a search until a proof (or disproof) were found or until we reached the practical limit of our search capabilities. The second type of problem arises when we are given a set of axioms and our task is that of generating consequences of those axioms. This type of problem may be reduced to the first by enumerating all possible consequences and seeking for a proof of each

in turn. More frequently, however, this second type of problem is handled by a process of enumerating proofs and listing the consequences of the proofs as theorems.

The most powerful techniques for theorem proving are based on the Resolution Principle, which was developed by J.A. Robinson (for a good survey of the work in this area, see [14]). The Resolution Principle is especially adapted for computer implementation, and it has little relationship to thought processes usually employed by logicians or mathematicians. However, it is extremely efficient and is now widely used in artificial intelligence. There are implementations corresponding to the two aspects mentioned above. That is, there are some implementations which are used to search for proofs for specified statements (in particular see [14]), while there are other implementations which are consequence generators (see [4], for an example). The Resolution Principle will be discussed in detail later.

Another important characteristic of "common sense" is the ability to generalize and form hypotheses on the basis of past experience. This ability is extremely important in the learning process. For example, after a child has been burned by a flame a very few times, he will make the generalization that fire produces pain. This generalization is not a deductive consequence of the past experience of the child, but rather it is an hypothesis based on the past experience. The ability to make such generalizations enables the child to avoid potentially dangerous situations and to capitalize on potentially pleasurable situations.

It remains an important factor throughout the entire life of the individual. We will refer to this pattern of reasoning as "induction".

There is some ambiguity surrounding the term "induction", as there are two distinct aspects of inductive reasoning. The first may be termed "discovery" and the second may be termed "evaluation". Given a set of data, we may simply be concerned with generating hypotheses which, if true, would account for the data; this is the discovery aspect. On the other hand, we may have a set of hypotheses and be concerned with determining which hypothesis is the best supported on the basis of available information (and perhaps how likely it is to be true); this is the evaluation aspect. In this thesis we will be almost completely concerned with the discovery aspect of inductive reasoning, and we will loosely use the term "induction" to refer to this aspect alone.

Inductive reasoning is usually thought to be most characteristic of the intellectual activity of the creative scientist and the detective. Both individuals have to look at a limited amount of data and make hypotheses, about the state of nature or about who committed the crime, respectively. However, inductive reasoning is by no means limited to such specialists. Our ability to anticipate the effects of our behaviour on our surroundings -- both physical objects and other individuals -- is dependent on our ability to reason inductively (see the discussion of induction in [9]). Any machine which would properly be described as intelligent would most certainly have to possess some ability to anticipate future events on the basis of its past experience,

and it would thus have to be able to make inductive inferences.

Inductive reasoning also appears to be an extremely important heuristic in making deductive inferences. Teachers of logic and mathematics frequently tell their students that when faced with a difficult theorem they are trying to prove, they should "reason backwards". That is, students are told to examine the theorem and attempt to discover an intermediate statement from which the theorem could be proved. Once such an intermediate is found, the original problem can be reduced to finding a proof of the intermediate. If the first intermediate is not easily provable, the student is told to search for a second intermediate from which the first is provable. This process continues until the problem is abandoned or until a proof is found. At each stage, the discovery of the intermediate statement is a matter of inductive inference. It is in this way that inductive inference can serve as a valuable heuristic in making very complicated deductive inferences. Thus inductive inference is important in the construction of intelligent machines not only as a device for learning and anticipating the future, but also as a device to promote swift and efficient deductive inferences.

The importance of using the "reasoning backward" heuristic in doing deductive proofs has long been recognized in artificial intelligence research. For example, the Logic Theorist used such procedures almost exclusively in its operation (see [11]). Inductive inference has also received attention in other areas of artificial intelligence. Heuristic DENDRAL is a program that combines both the discovery aspect

and the evaluation aspect of inductive reasoning in forming explanatory hypotheses in the area of organic chemistry (see [1]). The problem of discovering the phrase structure grammar for a language from a finite (usually short) list of expressions has received some attention (see the bibliography on grammatical induction in [2], p. 469). Work has also been done on the problem of determining a recursive function on the basis of a finite (usually small) set of input-output pairs (for one example, see [3]). Many researchers have commented on the importance of inductive inference for artificial intelligence (for example, see [4], p. 51; [2], pp. 446-449; and [16], pp. 175-178).

Thus far, no one has formulated a general theory of inductive inference for the language of first-order predicate calculus. (To some extent, first-order predicate calculus can be handled by the methods developed for grammars and for recursive function theory. However, there has been no formulation of the discovery aspect of inductive inference in predicate calculus.) Since the theory of deductive inference is couched almost exclusively in the predicate calculus, and since theorem proving devices make extensive use of this language, a really good "reasoning backward" heuristic program for deductive inferences will require a theory of inductive inference for the first-order predicate calculus. Further, the first-order predicate calculus is one of the most powerful formal languages we now have available for the representation of general problems. (For examples of the usefulness of predicate calculus in problem representation, see [12]. There are several types

of formal languages more powerful than first-order predicate calculus. Two particular types are higher-order calculi and modal logics. However, these languages are not in common use by researchers. Further, it appears that the methods presented here may be extended easily to these other languages.) Thus, if we wish to develop machines with the ability to reason inductively concerning a wide variety of material, one natural approach is to develop inductive inference techniques for that language.

In this thesis, a general theory of inductive inference for first-order predicate calculus is developed, and it is proved that the theory has the desired formal properties analogous to the properties of the theory of deductive inference. A simple method for converting (deductive) theorem generating programmes into (inductive) hypothesis generating programmes is presented. Finally, a method of using the inductive techniques to improve the efficiency of deductive theorem provers (with specific reference to the Resolution Principle) is discussed.

B. The Formal Language

Throughout the rest of this thesis, we will be talking about the first-order predicate calculus. A brief outline of the language will be given here. (More details concerning these technical matters may be found in standard logic texts. Two good sources are [8] and [17].) The following symbols plus right and left parentheses constitute the alphabet of the language:

$P_1^1, P_1^2, \dots, P_2^1, P_2^2, \dots$	predicates
$f_1^1, f_1^2, \dots, f_2^1, f_2^2, \dots$	functions
x_1, x_2, \dots	individual variables
a_1, a_2, \dots	individual constants
\vee	disjunction (inclusive)
$\&$	conjunction
\neg	negation
$(\forall x_1)$	universal quantifier
$(\exists x_1)$	existential quantifier

First we will outline the syntax of the language by means of the following definitions.

Definition 1: Term

- (a) a_i is a term, for every positive integer i .
- (b) x_i is a term, for every positive integer i .
- (c) If t_1, \dots, t_n are terms and f_i^n is a function, then $f_i^n(t_1, \dots, t_n)$ is a term, for every positive integer i .

Definition 2: If P_i^n is a predicate and t_1, \dots, t_n are terms, then $P_i^n t_1 \dots t_n$ is an atomic expression.

Definition 3: Expression

- (a) Every atomic expression is an expression.
- (b) If E_1 and E_2 are expressions, then so is $(E_1 \vee E_2)$.
- (c) If E_1 and E_2 are expressions, then so is $(E_1 \& E_2)$.
- (d) If E is an expression, then so is $\neg E$.

- (e) If E is an expression, then so is $(x_1)(E)$, for every positive integer i .
- (f) If E is an expression, then so is $(\exists x_1)(E)$, for every positive integer i .

Definition 4: The scope of a quantifier is the expression in parentheses which immediately follows the quantifier. Thus, for $(x_1)(E)$ and $(\exists x_1)(E)$, the scope of the quantifier in each case is the expression E .

Definition 5: If a variable x_1 occurs within an expression but not within the scope of a quantifier over that variable -- i.e., " (x_1) " or " $(\exists x_1)$ " -- then the variable is said to be free in that expression. A variable that is not free is said to be bound. A variable is bound by the first appropriate quantifier into whose scope it falls. Thus in the expression

$$(x_1)(P_1^1 x_1 \vee (\exists x_1)(P_2^1 x_1))$$

the " x_1 " occurring in " $P_1^1 x_1$ " is bound by " (x_1) ", while the " x_1 " occurring in " $P_2^1 x_1$ " is bound by " $(\exists x_1)$ " and not by " (x_1) ".

We will adopt several conventions with regard to the language. The set of all expressions will be referred to as " L ". We will frequently drop parentheses if there is no danger of confusion. We will assume that any expression containing free variables is to be preceded by universal quantifiers over those variables (in any order). For example, " $P_1^1 x_1$ " occurring by itself will be interpreted as " $(x_1)P_1^1 x_1$ ". If the variable occurring in a quantifier is not free in the scope of that

quantifier, then the quantifier will be ignored. It should be clear that E, E_1, E_2, \dots are being used as meta-expressions to refer to the expressions of L . Further, we will ignore all connectives other than " \forall ", " $\&$ ", and " \neg " since the others may be defined in terms of these.

We will now give a brief sketch of the formal semantic theory for L . The intuitive notion behind the semantics is that of interpretation. Essentially what is done in providing an interpretation for L is the specification of a non-empty domain of discourse and a method of determining the "meaning" (and hence truth value) of every expression of L .

An interpretation, say I , for L is an ordered pair, say (D, V) , where D is a set of objects (e.g. numbers), and V is a valuation function. Each individual constant, a_i , is assigned a unique value, $V(a_i)$, in D . Each n -ary function symbol, f_i^n , is assigned an n -ary function, $V(f_i^n)$ from D^n into D . Terms which do not contain variables will then be assigned values automatically depending on the values assigned to the constants and the functions. The value of a_i is just $V(a_i)$, and the value of $f_i^n(t_1, \dots, t_n)$ is given inductively by $V(f_i^n)(V(t_1), \dots, V(t_n))$. Thus all terms with no variables receive an interpretation as some member of D . Variables and terms containing variables will receive no definite assignment but will take on various values in D , as will be indicated below. Each predicate symbol, P_i^n , is assigned to an n -ary relation, $V(P_i^n)$, on D .

Intuitively, an atomic expression with no free variables, say

$P_i^n t_1 \dots t_n$, is interpreted as asserting that the relation $V(P_i^n)$ holds for the ordered n -tuple $(V(t_1), \dots, V(t_n))$. More complicated expressions are similarly interpreted. An expression is then true under a given interpretation just in case the assertion it is interpreted to be making is true. There is no loss of generality in assuming that in any expression, no two quantifiers contain the same variable. We may then make these ideas more precise by the following:

(V1) If t_1, \dots, t_n are terms all of which have been assigned a value in D by V , then:

$$\begin{aligned} V(P_i^n t_1 \dots t_n) &= \text{true if } (V(t_1), \dots, V(t_n)) \in V(P_i^n) \\ &= \text{false otherwise} \end{aligned}$$

(V2) $V(\neg E) = \text{true if } V(E) = \text{false}$
 $= \text{false otherwise}$

(V3) $V(E_1 \vee E_2) = \text{true if } V(E_1) = \text{true or } V(E_2) = \text{true}$
 $= \text{false otherwise}$

(V4) $V(E_1 \& E_2) = \text{true if } V(E_1) = \text{true and } V(E_2) = \text{true}$
 $= \text{false otherwise}$

(V5) $V((\exists x_1)E) = \text{true if there is some } d \in D \text{ such that:}$
 $\text{if we assign } V(x_1) \text{ the value } d \text{ then } V(E) = \text{true}$
 $= \text{false otherwise}$

(V6) $V((\forall x_1)E) = \text{true if for every } d \in D: \text{ if we assign}$
 $V(x_1) \text{ the value } d \text{ then } V(E) = \text{true}$
 $= \text{false otherwise}$

Note that expressions with free variables are to be interpreted as being universally quantified over those variables.

We will say that an expression E is logically true (LT) if and only if in every possible interpretation, $V(E) = \text{true}$. An expression E will be said to be logically false (LF) if and only if in every possible interpretation, $V(E) = \text{false}$. An expression that is neither LF nor LT will be said to be logically contingent (LC). Two expressions E_1 and E_2 will be said to be logically equivalent (LE) if and only if in every possible interpretation, $V(E_1) = V(E_2)$. We will say that an expression E is satisfiable if and only if there is some interpretation in which E is true. We will say that an expression E is falsifiable if and only if there is some interpretation in which E is false. An expression is said to be satisfied by an interpretation in which it is true and is said to be falsified by an interpretation in which it is false. We will use "formula" interchangeably with "expression".

It is not difficult to specify a formal syntactical theory of deductive inference for L , and many such theories are known. There are in general two types of such theories. One type consists of a set of axioms, all LT, and a set of inference rules; such systems are called axiomatic systems. The other type consists of only a set of inference rules; such systems are called natural deduction systems. In this thesis we will deal almost exclusively with natural deduction systems, although the extension of the theory to be presented to axiomatic systems is quite simple.

If one expression E_2 is derivable by the inference rules from another expression E_1 , we will write $E_1 \vdash E_2$. (If we wish to make explicit which set of inference rules we are using, we will attach an appropriate subscript to the symbol " \vdash ".) Similarly, if λ is a set of expressions from which the expression E is derivable, we will write $\lambda \vdash E$. The standard rules of deductive inference are truth preserving in the sense that if $\lambda \vdash E$, then in any interpretation under which all of the expressions in λ take the value true, E also takes the value true. Further, the sets of deductive inference rules with which we will be concerned also have the property of completeness. That is, for any set of expressions λ and any expression E , if λ and E are such that in every interpretation in which every member of λ is true E is also true, then $\lambda \vdash E$. Finally, the deductive rules are consistent in the sense that if λ is a set of expressions such that there is some interpretation in which every member of λ is true, then there is no expression E such that both $\lambda \vdash E$ and $\lambda \vdash \neg E$.

Since the theory to be developed is completely general, no specific deductive system will be presented at this time. Later the system based on the Resolution Principle will be considered in some detail.

Chapter II: F-Rules*

A. Preliminary Comments and Assumptions

Bernard Meltzer has recently suggested that it is possible to obtain a system of inductive logic that is "relatively complete" for languages with the syntactical structure of the first-order predicate calculus. (See [7].) Part of the procedure which he outlines depends on "inverting" the deductive inference rules which may be specified for such languages. We will here outline an alternative procedure for obtaining the same results. Our procedure has several advantages. First, it makes the formal proof of the desired results easier to obtain. Second, it gives a set of rules which closely parallel the normal deductive rules, and the rules are for that reason more intuitive to human users. Third, it provides an easy method for turning theorem generating devices into hypothesis generating devices.

Throughout the rest of the chapter, we will assume that there is given a natural deduction system PCT (predicate calculus for truths) of deduction for L . Such a system will consist of a set of truth preserving rules TR_1, \dots, TR_m . If an expression E_2 is derivable by the rules from an expression E_1 , we will write $E_1 \vdash_T E_2$. Similarly, for any set of expressions λ , possibly empty, if the expression E is derivable in PCT from the expressions in λ , we will write $\lambda \vdash_T E$. We will assume

* A version of this chapter has been published; see [10].

that the deductive system is complete and consistent. We will also assume that given any ordered string of expressions of L , say E_1, \dots, E_p , there is an effective procedure for determining whether or not the sequence constitutes a valid proof of E_p in PCT.

B. Some Difficulties with Meltzer's Procedure

Meltzer states:

If now, for example, our language were that of first-order predicate calculus, we could take some complete deductive system and invert all its rules. Since the original rules are effective, the inverted ones are too. And the resulting system, applied to premises A , would yield all the hypotheses which deductively imply A , and no others. ([7], p. 190)

For example, consider the following rule: "From any two expressions of the form ' E_1 ' and ' $\neg E_1 \vee E_2$ ', infer ' E_2 ' ". For the language we are considering, this rule corresponds to the standard rule of modus ponens. Strictly speaking, the "inverse" of this rule would read, "From an expression ' E_2 ', infer any two expressions of the form ' E_1 ' and ' $\neg E_1 \vee E_2$ ' ". This latter rule is unsatisfactory because it seems to allow us to infer (in part) any expression ' E_1 ' from any other expression ' E_2 ', while the reverse inference is not a valid deductive inference. A little more needs to be said about inverting the deductive rules. One way to avoid this problem is to require that the "inverse" rules have single expressions as consequences. Thus, an acceptable version of the inverse of the above rule would be, "From any expression ' E_2 ', infer an expression of the form ' $E_1 \& (\neg E_1 \vee E_2)$ ' where ' E_1 ' is any expression

at all".

The difficulty to which we have alluded derives from the following points. Our rules of deductive inference are frequently given in the form of many-to-one relations. That is, they usually say something like, "From expressions of the form ' E_1 ', ..., ' E_r ', infer an expression of the form ' E_q ' ". We will call the $E_1 - E_r$ the antecedents of the rule and E_q the consequent of the rule. (In some cases, $r = 1$ and the statement of the rule has the form of a one-to-one relation.) Even a rule like 'addition' — "From an expression ' E_1 ', infer an expression of the form ' $E_1 \vee E_2$ ', where ' E_2 ' is any expression at all" — which allows an infinite number of consequences is stated in the form of a many-to-one (actually one-to-one) relation.

The deductive rules have the property of being truth preserving. That is, if the antecedents in the rule are all true, then the inferential consequents of the rule are also true. (This statement is somewhat of an oversimplification. Some rules which may be included, for example the rule of conditional proof which is presented below, are not of the simple form here discussed. They are nevertheless truth preserving in that if they are used to derive a conclusion from a set of premises and the premises are true, then the conclusion must also be true.) However, we must be careful if we attempt to reason in the other direction. We cannot say that if the consequent of a rule is false then all of the antecedents of the rule are also false. We can only say that if the consequent of the rule is false, then the conjunction of all of the

antecedents of the rule is also false. Thus to be correctly stated the inverse rules must direct inference from the consequent to the conjunction of the antecedents. An alternative procedure which is much more complicated is to direct inference from the consequent of a deductive rule to the individual antecedents of that rule and "flag" the antecedents to indicate that they must be regarded collectively. Both of these methods seem to destroy the symmetry between the deductive and inductive inferences on which Meltzer's discussion is based.

Another difficulty arises when one considers certain complicated, but highly useful deductive rules. For example, a rule of conditional proof may be stated as follows:

At any stage of a proof, say line r , any premise E may be introduced, providing that:

- (a) the premise is discharged at or before the last line of the proof, say at line n , by writing a line of the form ' $\neg E \vee Q$ ', where ' Q ' is the n -1st line of the proof, and appealing to lines r and $n-1$; and
- (b) if the proof proceeds past the line on which the introduced premise was discharged, line n , then no line past the n th may appeal to any of the lines r through $n-1$, where r is the line at which the premise was introduced.

There does not seem to be any way of "inverting" such a rule. However, this rule is widely used and usually shortens proofs considerably. At the very least, an inversion of such a rule would be very complicated. Given an arbitrary string of expressions E_1, \dots, E_p , we could check to see if the reverse string E_p, \dots, E_1 makes use of the above rule, and perhaps in this way check the original string to see if it makes use of the "inverse" of the above rule. But the problem is that we still have no idea what the "inverse" rule is, nor how to sequentially (front to back, top to bottom) construct strings according to the "inverse" rule.

C. Induction and Falsehood Preserving Rules

Consider the problem of determining the consequences which follow from a given set of axioms. Using the technique of Gödel numbering it is possible to assign to each proof a unique natural number. Further, given an arbitrary natural number it is possible to determine whether or not it is a number assigned to a valid proof, and if so, it is possible to reconstruct the proof. To obtain the consequences of an arbitrary set of axioms, we start with 1 and go through the natural numbers in order. We simply ask of each in turn if it is the number assigned to a valid proof from the given axioms. If it is not, we ignore it and go to the next number. If it is, we simply put the last line of the corresponding proof on our list of theorems, and then go on to the next number. We could continue this process forever, our list of theorems growing larger and larger. But for any statement E which is a

consequence of the given axioms, E would eventually be put on the list.

This method has certain drawbacks that make it impractical to use, for the most part. The calculations involved rapidly become difficult and complex due to the extremely large numbers one must examine. The length of time required for one to determine whether or not a number with one billion digits corresponds to a valid proof may very well be longer than a normal life span. Thus in actual practice, various heuristic procedures which do not guarantee solutions are used; such procedures usually involve the search of proof trees including pruning devices. Nevertheless, the practical difficulties involved do not detract from the fact that there is an effective procedure for generating the consequences of a given axiom set.

Suppose now that we have a set of rules that preserves falsehood. We will call such rules "f-rules". If an expression E_2 is obtainable from an expression E_1 by such rules, then it could be stated that if E_1 is false then E_2 must also be false. In more technical terms, E_2 is false under every interpretation which makes E_1 false. We will call the system based on such rules PCF (predicate calculus for falsehoods). Let λ be a set of expressions of L , perhaps empty. If there is a proof of expression E in PCF from the set of expressions in λ , we will write $\lambda \vdash_F E$. Similarly, we will write $E_1 \vdash_F E_2$ where expression E_2 is obtainable from expression E_1 in PCF.

A set of f-rules has a very interesting property for the purposes of this discussion. Suppose E_2 is obtainable from E_1 by a set of

f-rules. Then it turns out that E_1 can be deduced from E_2 by any complete set of truth preserving rules. To see that this is so, note that since E_2 is obtainable from E_1 by the f-rules, we know that if E_1 is false then E_2 must be false. But this is equivalent to saying that if E_2 is not false then E_1 is not false. But this simply means that if E_2 is true then E_1 is true. But then E_1 is derivable from E_2 by any complete set of truth preserving rules. The importance of this fact about f-rules is readily apparent when one notes that this property of the rules means that falsehood preserving rules lead from consequences to axioms, or in the empirical realm, from statements of facts to hypotheses. Thus we have shown the following:

Theorem 1 (Theorem of f-rules): If an expression E_2 is obtainable from an expression E_1 by falsehood preserving rules, then E_1 is obtainable from E_2 by a complete set of truth preserving rules.

We will say that the set of f-rules is complete if and only if for any two expressions E_1 and E_2 , if it is the case that E_2 is false whenever E_1 is false, then E_2 can be obtained from E_1 by the f-rules. Now consider some given statement E , describing some matter of fact, and let T be any arbitrary theory from which E may be deduced by truth preserving rules. Then if T is true, E is true. But this means that if E is false then T is false. Thus T may be obtained from E by a complete set of f-rules. Similarly, if we wanted a theory to account for several facts, say U , V , and W , then we could take as an axiom the conjunction $U \& V \& W$

and apply the f-rules as above to obtain the desired theory.

Now, suppose the f-rules are of a rather simple kind, like the truth preserving rules. That is, suppose there is an effective procedure for determining whether or not one expression follows from a given finite set of expressions by one application of any one of the rules. Then we can employ the technique of Gödel numbering again. To each f-proof (We will call a proof using the f-rules an f-proof.) we may assign a unique natural number. Given any natural number we may determine whether or not it is the number of a proof, and if so we may reconstruct the proof. Using the same technique outlined above for the truth preserving rules, we may generate all of the f-consequences of any given statement. But any theory T from which a given expression E is deducible (using truth preserving rules) is an f-consequence of E (assuming our set of f-rules is complete). Thus the procedure would generate T. For example, if Kepler's laws are deducible using a set of truth preserving rules from Newtonian mechanics, then using the procedure outlined above, we could start with Kepler's laws and eventually generate Newtonian mechanics.

D. Obtaining Falsehood Preserving Rules

The discussion thus far has been somewhat hypothetical in that we have not yet indicated how one would go about obtaining a set of f-rules. We will now proceed to outline a procedure for obtaining a set of f-rules from any standard set of truth preserving rules. Further, the set of f-rules will be complete if the set of truth preserving rules from

which they are derived is complete. Since the existence of complete sets of truth preserving rules is well known, this will complete the proof that the theories from which a given expression follows may be generated. Further, by supplying a set of truth preserving rules and following the proof, one may obtain the desired generation procedure, and hence the proof is constructive.

The function below will be used to operate on expressions in the object language. Let F be defined recursively as follows:

$F(E) = E$, if E is any atomic expression

$F(\neg E) = \neg F(E)$, for any expression E

$F(E_1 \vee E_2) = F(E_1) \& F(E_2)$, for any expressions E_1 and E_2

$F(E_1 \& E_2) = F(E_1) \vee F(E_2)$, for any expressions E_1 and E_2

$F((x_i)E) = (\exists x_i)F(E)$, for any expression E and any variable x_i

$F((\exists x_i)E) = (x_i)F(E)$, for any expression E and any variable x_i

Consider an expression of the object language, for example

$'(x_1)(P_{31}^1 \vee \neg P_{41}^1)'$. Applying the above definition, the F transform of the expression is $'(\exists x_1)(P_{31}^1 \& \neg P_{41}^1)'$.

To obtain a set of f -rules, it is only necessary to take the F transform of the truth preserving rules. This is done by taking the F transform of every expression mentioned in the rules, being concerned

only with changes brought about in the connectives and ignoring 'F' when it is applied to expression letters. A rule of substitution would be unchanged, since it does not explicitly mention any of the connectives. The F transform of the rule of conditional proof stated in section B, above, would read exactly the same except that in condition (a), the expression ' $\neg E \vee Q$ ' would be replaced by ' $\neg E \& Q$ '. The F transform of the modus ponens rule of section B, above, would read, "From any two expressions of the form ' E_1 ' and ' $\neg E_1 \& E_2$ ', infer ' E_2 ' ". These examples should make the procedure clear. We will refer to the f-rules obtained in this way as FR_1, \dots, FR_m , corresponding to TR_1, \dots, TR_m , respectively.

We now need to argue that the rules obtained as outlined above are indeed f-rules, i.e., that they are falsehood preserving. We also need to show that the f-rules are complete if the truth preserving rules are. Finally, we need to show that the f-rules are consistent if the truth preserving rules are. In order to do so, we will need some intermediate results.

Theorem 2: For any expression E , $F(F(E)) = E$.

Proof: By induction on the number n of connectives and quantifiers in E . Suppose $n = 0$. Then E is atomic. But then by the definition of F :

$$F(F(E)) = F(E) = E$$

Now, suppose that for all $n < p$, where $p > 0$, the theorem is true of all

expressions of length n . We must show that the theorem holds for $n = p$.

case 1: $E = \neg E_1$. Then:

$$F(F(E)) = F(F(\neg E_1)) = F(\neg F(E_1)) = \neg F(F(E_1))$$

But by the induction hypothesis, $F(F(E_1)) = E_1$. Hence $F(F(E)) = \neg E_1 = E$.

case 2: $E = E_1 \vee E_2$. Then:

$$F(F(E)) = F(F(E_1 \vee E_2)) = F(F(E_1) \& F(E_2)) = F(F(E_1)) \vee F(F(E_2))$$

But by the induction hypothesis, $F(F(E_1)) = E_1$ and $F(F(E_2)) = E_2$. Thus,

$$F(F(E)) = E_1 \vee E_2 = E.$$

case 3: $E = E_1 \& E_2$. This case is similar to case 2, exchanging ' $\&$ ' and ' \vee ' throughout.

case 4: $E = (x_1)E_1$. Then:

$$F(F(E)) = F(F((x_1)E_1)) = F((\exists x_1)F(E_1)) = (x_1)F(F(E_1))$$

By the induction hypothesis, $F(F(E_1)) = E_1$. Hence $F(F(E)) = (x_1)E_1 = E$.

case 5: $E = (\exists x_1)E_1$. This case is similar to case 4, exchanging ' (x_1) ' and ' $(\exists x_1)$ ' throughout.

Thus the theorem holds for $n = p$, and hence by induction it holds for all n . Q.E.D.

Let the function T be defined as follows:

$$T(E) = \neg E, \text{ if } E \text{ is any atomic expression}$$

$$T(\neg E) = \neg T(E), \text{ for any expression } E$$

$$T(E_1 \vee E_2) = T(E_1) \vee T(E_2), \text{ for any expression } E_1$$

$$\text{and } E_2$$

$$T(E_1 \& E_2) = T(E_1) \& T(E_2) , \text{ for any expressions } E_1 \\ \text{and } E_2$$

$$T((x_1)E) = (x_1)T(E) , \text{ for any expression } E \text{ and any} \\ \text{variable } x_1$$

$$T((\exists x_1)E) = (\exists x_1)T(E) , \text{ for any expression } E \text{ and any} \\ \text{variable } x_1$$

The function T simply replaces any atomic component of an expression by the negation of that component. The proof of the following theorem about T is not difficult, depending only on certain semantical notions. However, it is not relevant to our concerns here, so it is omitted.

Theorem 3: Let E be any expression of L . Then E is LT if and only if $T(E)$ is LT. Similarly, E is LF if and only if $T(E)$ is LF. Finally, E is LC if and only if $T(E)$ is LC.

We may now prove an important result concerning the relation of the F transform of an expression to the T transform of the same expression. This relation will be used in the proof of the succeeding theorem.

Theorem 4: For any expression E , $F(E)$ is LE to $\neg T(E)$.

Proof: The proof is by induction on the number n of connectives and quantifiers in E . Suppose $n = 0$; then E is an atomic expression. Thus, $F(E) = E$, and $T(E) = \neg E$; clearly E is LE to $\neg \neg E$. Now, suppose the theorem holds for all $n < p$, where $p > 0$. We must show

that the theorem holds for $n = p$.

case 1: $E = \neg E_1$. Then $F(E) = \neg F(E_1)$. By the induction hypothesis, $F(E_1)$ is LE to $\neg T(E_1)$. Hence $F(E)$ is LE to $\neg \neg T(E_1)$. But $\neg \neg T(E_1) = \neg T(\neg E_1)$. Hence $F(E)$ is LE to $\neg T(E)$.

case 2: $E = E_1 \vee E_2$. Then $F(E) = F(E_1) \& F(E_2)$. By the induction hypothesis, $F(E_1)$ is LE to $\neg T(E_1)$, and $F(E_2)$ is LE to $\neg T(E_2)$. Hence $F(E)$ is LE to $\neg T(E_1) \& \neg T(E_2)$. But,

$$\neg T(E_1) \& \neg T(E_2) \text{ LE } \neg(T(E_1) \vee T(E_2)) = \neg T(E_1 \vee E_2)$$

Thus, $F(E)$ is LE to $\neg T(E)$.

case 3: $E = E_1 \& E_2$. This case is similar to case 2, exchanging ' $\&$ ' and ' \vee ' throughout.

case 4: $E = (x_1)E_1$. Then $F(E) = (\exists x_1)F(E_1)$. By the induction hypothesis, $F(E_1)$ is LE to $\neg T(E_1)$. Further, neither F nor T adds, removes, nor exchanges variables. Hence $F(E)$ is LE to $(\exists x_1) \neg T(E_1)$. But $(\exists x_1) \neg T(E_1)$ is LE to $\neg (x_1)T(E_1)$. Further, $\neg (x_1)T(E_1) = \neg T((x_1)E_1)$. Hence $F(E)$ is LE to $\neg T(E)$.

case 5: $E = (\exists x_1)E_1$. This case is similar to case 4, exchanging ' $(\exists x_1)$ ' and ' (x_1) ' throughout.

Thus the theorem holds for $n = p$, and hence by induction it holds for all n . Q.E.D.

Theorem 5: For any expression E , E is LT if and only if $F(E)$ is LF; E is LF if and only if $F(E)$ is LT; E is LC if and only if $F(E)$ is LC.

Proof: By Theorem 3, E is LT (LF or LC) if and only if $T(E)$ is LT (LF or LC). By Theorem 4, $F(E)$ is LE to $\neg T(E)$. But for any expression E_1 , we have the following:

(i) E_1 is LT if and only if $\neg E_1$ is LF.

(ii) E_1 is LF if and only if $\neg E_1$ is LT.

(iii) E_1 is LC if and only if $\neg E_1$ is LC.

Thus E is LT (LF or LC) if and only if $\neg T(E)$ is LF (LT or LC); and the latter holds if and only if $F(E)$ is LF (LT or LC). Q.E.D.

In the following material, S_1, S_2, \dots will be used as meta-meta-expressions. The same connectives and quantifiers will be used below for the object language, meta-language, and meta-meta-language expressions, but this should occasion no confusion.

Lemma 1: An object language expression E_1 has the form specified by a meta-language expression S_1 if and only if $F(E_1)$ has the form specified by the F transform of S_1 , where the F transform of S_1 is obtained by taking $F(S_1)$, being concerned only with changes brought about in the connectives and quantifiers, and ignoring 'F' when it is applied to an expression letter.

Proof: The proof is by an easy induction on the number n of connectives and quantifiers in S_1 . Suppose $n = 0$; then S_1 is simply an expression letter. Thus the F transform of S_1 is just an expression letter. But any syntactically well-formed object language expression has the form specified by a single expression letter of the meta-language.

Hence, the lemma holds for $n = 0$. Now, suppose the theorem holds for all $n < p$, where $p > 0$. We must show that the theorem holds for p . We will have five cases as in the proof of Theorem 3. In each case, the theorem follows immediately from the induction hypothesis and the definition of F . Thus the lemma holds for $n = p$, and hence by induction it holds for all n . Q.E.D.

Theorem 6: Let λ be a set of expressions of L , and let $F(\lambda)$ be the set of expressions whose members are the F transforms of the members of λ . Then $\lambda \vdash_T E$ if and only if $F(\lambda) \vdash_F F(E)$, where E is any expression of L .

Proof: Suppose $\lambda \vdash_T E$. Let E_1, \dots, E_p, E be a series of expressions of L , and let $F(E_1), \dots, F(E_p), F(E)$ be a series of expressions of L obtained from the first by taking the F transform of each expression in that series. Then we will show that the first series constitutes a proof of E in PCT if and only if the second series constitutes a proof of $F(E)$ in PCF. Consider an arbitrary step E_i in the series. There are two cases:

case 1: E_i is an axiom. But E_i is an axiom (in λ) if and only if $F(E_i)$ is an axiom (in $F(\lambda)$).

case 2: E_i follows according to rule TR_j from E_{i1}, \dots, E_{it} . Since the function F changes no variables, constants, or predicates, restrictions on variables, constants, and predicates are not changed by taking the F transform of the rules. For example, if we require for the

application of rule TR_j that the variable x_r should not have been introduced at an earlier stage by rule TR_k , then FR_j will require for its application that the variable x_r should not have been introduced at an earlier stage by the rule FR_k . Thus E_i satisfies such restrictions with regard to TR_j if and only if $F(E_i)$ satisfies such restrictions with regard to FR_j . Now, by Lemma 1, $E_1, E_{11}, \dots, E_{it}$ have the syntactical forms specified by TR_j if and only if $F(E_1), F(E_{11}), \dots, F(E_{it})$ have the syntactical forms specified by FR_j . Thus, E_i follows according to rule TR_j if and only if $F(E_i)$ follows according to FR_j .

Thus each step in the E_i series is justified if and only if each step in the $F(E_i)$ series is justified. Thus the E_i series constitutes a proof of E in PCT if and only if the $F(E_i)$ series constitutes a proof of $F(E)$ in PCF. Similarly, if we begin with the supposition that $F(\lambda) \vdash_F F(E)$, we can obtain $\lambda \vdash_T E$. Hence, $\lambda \vdash_T E$ if and only if $F(\lambda) \vdash_F F(E)$. Q.E.D.

We may now prove the three important results mentioned above. We will begin by stating the corresponding results which are assumed to hold for PCT.

Theorem 7: The rules of PCT are truth preserving in the sense that if $\{E_1, \dots, E_p\} \vdash_T E$ and all of the E_i are true, where $1 \leq i \leq p$, then E is also true.

Theorem 8: The rules of PCT are complete in the sense that for any set of expressions, say $\{E_1, \dots, E_p\}$, and any expression E , if E is true in every interpretation in which all the E_i are true, where $1 \leq i \leq p$, then $\{E_1, \dots, E_p\} \vdash_T E$.

Theorem 9: The rules of PCT are consistent in the sense that for any set of expressions, say $\{E_1, \dots, E_p\}$, if there is an interpretation in which all of the E_i are true, where $1 \leq i \leq p$, then there is no expression E such that both $\{E_1, \dots, E_p\} \vdash_T E$ and $\{E_1, \dots, E_p\} \vdash_T \neg E$.

Theorem 10: The rules of PCF are falsehood preserving in the sense that if $\{E_1, \dots, E_p\} \vdash_F E$ and all of the E_i are false, where $1 \leq i \leq p$, then E is also false.

Proof: Suppose $\{E_1, \dots, E_p\} \vdash_F E$. By Theorem 2, $\{F(F(E_1)), \dots, F(F(E_p))\} \vdash_F F(F(E))$. By Theorem 6, $\{F(E_1), \dots, F(E_p)\} \vdash_T F(E)$. By Theorem 7, if all of the $F(E_i)$ are true, where $1 \leq i \leq p$, then $F(E)$ is also true. But this means that the following expression is LT:

$$(1) \quad \neg F(E_1) \vee \dots \vee \neg F(E_p) \vee F(E)$$

But then by Theorem 5, the F transform of (1) is LF. The F transform of (1) is given by:

$$(2) \quad \neg F(F(E_1)) \& \dots \& \neg F(F(E_p)) \& F(F(E))$$

By Theorem 2, (2) is the same as:

$$(3) \quad \neg E_1 \& \dots \& \neg E_p \& E$$

But the fact that (3) is LF means that if all of the E_i are false, then E must be false as well. Thus the rules of PCF are falsehood preserving. Q.E.D.

Theorem 11: The rules of PCF are complete in the sense that for any set of expressions, say $\{E_1, \dots, E_p\}$, and any expression E , if E is false in every interpretation in which all the E_i are false, where $1 \leq i \leq p$, then $\{E_1, \dots, E_p\} \vdash_F E$.

Proof: Suppose E is false in every interpretation in which all of the E_i are false, for $1 \leq i \leq p$. Then the following expression must be LF.

$$(1) \quad \neg E_1 \ \& \ \dots \ \& \ \neg E_p \ \& \ E$$

By Theorem 5, the F transform of this expression must be LT. The F transform of (1) is given by:

$$(2) \quad \neg F(E_1) \vee \dots \vee \neg F(E_p) \vee F(E)$$

But this means that $F(E)$ is true in every interpretation in which all of the $F(E_i)$ are true, for $1 \leq i \leq p$. Thus by Theorem 8:

$$(3) \quad \{F(E_1), \dots, F(E_p)\} \vdash_T F(E)$$

From (3) we obtain by Theorem 6:

$$(4) \quad \{F(F(E_1)), \dots, F(F(E_p))\} \vdash_F F(F(E))$$

From (4) we obtain by Theorem 2:

$$(5) \quad \{E_1, \dots, E_p\} \vdash_F E$$

Q.E.D.

Theorem 12: The rules of PCF are consistent in the sense that for any set of expressions, say $\{E_1, \dots, E_p\}$, if there is an interpretation in which all of the E_i are false, for $1 \leq i \leq p$, then there is no expression E such that both $\{E_1, \dots, E_p\} \vdash_F E$ and $\{E_1, \dots, E_p\} \vdash_F \neg E$.

Proof: Suppose there is an interpretation such that all of the E_i are false, for $1 \leq i \leq p$. Then $E_1 \vee \dots \vee E_p$ is not LT. Then by Theorem 5 and the definition of F , $F(E_1) \& \dots \& F(E_p)$ is not LF. Thus there is an interpretation making all of the $F(E_i)$ true, for $1 \leq i \leq p$. Now, suppose contrary to the theorem that for some E , $\{E_1, \dots, E_p\} \vdash_F E$ and $\{E_1, \dots, E_p\} \vdash_F \neg E$. Then by Theorem 2 and the definition of F , $\{F(F(E_1)), \dots, F(F(E_p))\} \vdash_F F(F(E))$ and $\{F(F(E_1)), \dots, F(F(E_p))\} \vdash_F F(\neg F(E))$. By Theorem 6, $\{F(E_1), \dots, F(E_p)\} \vdash_T F(E)$ and $\{F(E_1), \dots, F(E_p)\} \vdash_F \neg F(E)$. But this contradicts Theorem 9, and hence the theorem is proved. Q.E.D.

E. Advantages

In section C, it was demonstrated that a complete set of falsehood preserving rules could be used to generate all of the hypotheses from which a given expression could be deduced. In section D, a method for obtaining such a set of rules was described, and it was proved that the set of rules obtained by the indicated method would be falsehood preserving, complete (in the sense of Theorem 11), and consistent. We will now discuss a few advantages of sets of rules so obtained.

In addition to allowing relatively simple proofs of the desired theorems (see section D), the method generates rules that are somewhat intuitive to use. If one is used to a particular set of deductive rules, then he should find the derived inductive rules rather familiar. First of all, there will be one derived rule for each of the original rules. Secondly, each derived rule will have a structure very similar to that of the corresponding original rule. For example, compare the rule of conditional proof in section B with its F transform.

Another advantage of the procedure described above stems from Theorems 2 and 6 and the Theorem of f -rules. Suppose we have a consequence generating machine M . That is, suppose that when given a set of axioms as input, M begins to generate the deductive consequences of those axioms. It is easy to turn M into an hypothesis generating machine. Suppose we have a set of data statements E_1, \dots, E_p and that we would like an hypothesis to account for the E_i . First, form the conjunction of the E_i ; let $E = E_1 \& \dots \& E_p$. Now feed input $F(E)$ into machine M , and suppose M outputs an expression R . (R is an arbitrary one of the resulting outputs.) Then because M is a deductive consequence generator, we have $F(E) \vdash_T R$. By Theorem 6, $F(F(E)) \vdash_F F(R)$. By Theorem 2, $E \vdash_F F(R)$. But by the Theorem of f -rules, we have $F(R) \vdash_T E$. That is, to obtain hypotheses which entail (deductively) E , we feed M the expression $F(E)$. Hypotheses which entail E are obtained by taking the F transforms of the outputs of M . In this way, a consequence generator can be turned into an hypothesis generator.

Chapter III: Resolution

A. Deductive Resolution

In this section we will give an account of a specific system of deductive inference which was designed primarily for machine implementation. The system, which is simply called the Resolution Principle, was originally proposed by J.A. Robinson but has been extensively developed by others. (For the original formulation, see [13]. We will also make extensive use of the development given in [5].) We will not discuss these later developments.

In order to apply the Resolution Principle to a set of expressions, we must first put those expressions in a special form which will here be referred to as (universal) quantifier free conjunctive normal form (henceforth, (u)qfcnf). We will first discuss how to place any given expression into (u)qfcnf. If we are working with a set of expressions $\{E_1, \dots, E_n\}$ rather than a single expression, the method to be outlined below is to be applied to the conjunction of the members of the set, i.e., $E_1 \& \dots \& E_n$. Thus from now on we will assume that we are dealing with a single expression. Further, it is assumed that the expression initially contains no free variables. Free variables are to be regarded as being bound by universal quantifiers in accordance with the conventions of chapter I. We may also assume that all vacuous quantifiers have been removed. That is, if an expression $(\exists x_1)E$ or $(x_1)E$ occurs as part of the expression under consideration, where E does

not have any free occurrence of x_1 , then the quantifier is simply removed.

The following replacement rules will be used. Since they express logical equivalences (indicated by "LE"), we may replace expressions on the left by expressions on the right, and vice versa, with no change in truth value of the original expression.

$$(A1) \quad \neg (E_1 \& E_2) \text{ LE } \neg E_1 \vee \neg E_2$$

$$(A2) \quad \neg (E_1 \vee E_2) \text{ LE } \neg E_1 \& \neg E_2$$

$$(A3) \quad \neg \neg E \text{ LE } E$$

$$(A4) \quad \neg (x_1)E \text{ LE } (\exists x_1) \neg E$$

$$(A5) \quad \neg (\exists x_1)E \text{ LE } (x_1) \neg E$$

$$(A6) \quad \text{If } E_2 \text{ has no free occurrences of } x_1, \text{ then}$$

$$(x_1)E_1 \& E_2 \text{ LE } (x_1)(E_1 \& E_2) \text{ , and}$$

$$E_2 \& (x_1)E_1 \text{ LE } (x_1)(E_2 \& E_1)$$

$$(A7) \quad \text{If } E_2 \text{ has no free occurrences of } x_1, \text{ then}$$

$$(x_1)E_1 \vee E_2 \text{ LE } (x_1)(E_1 \vee E_2) \text{ , and}$$

$$E_2 \vee (x_1)E_1 \text{ LE } (x_1)(E_2 \vee E_1)$$

$$(A8) \quad \text{If } E_2 \text{ has no free occurrences of } x_1, \text{ then}$$

$$(\exists x_1)E_1 \& E_2 \text{ LE } (\exists x_1)(E_1 \& E_2) \text{ , and}$$

$$E_2 \& (\exists x_1)E_1 \text{ LE } (\exists x_1)(E_2 \& E_1)$$

$$(A9) \quad \text{If } E_2 \text{ has no free occurrences of } x_1, \text{ then}$$

$$(\exists x_1)E_1 \vee E_2 \text{ LE } (\exists x_1)(E_1 \vee E_2) \text{ , and}$$

$$E_2 \vee (\exists x_1)E_1 \text{ LE } (\exists x_1)(E_2 \vee E_1)$$

$$\begin{aligned}
(A10) \quad & E_1 \& (E_2 \vee E_3) \text{ LE } (E_1 \& E_2) \vee (E_1 \& E_3), \text{ and} \\
& (E_1 \vee E_2) \& E_3 \text{ LE } (E_1 \& E_3) \vee (E_2 \& E_3) \\
(A11) \quad & E_1 \vee (E_2 \& E_3) \text{ LE } (E_1 \vee E_2) \& (E_1 \vee E_3), \text{ and} \\
& (E_1 \& E_2) \vee E_3 \text{ LE } (E_1 \vee E_3) \& (E_2 \vee E_3)
\end{aligned}$$

The application of the rules is not restricted to the entire expression under consideration; rather, they may also be applied to sub-expressions. For example, the expression $(x_1)(E_1 \& E_2)$ is LE to $(x_1)\neg(\neg E_1 \vee \neg E_2)$ by an application of rules A1 and A3.

The first step in reducing an expression E to (u)qfcnf is to reduce the scope of each negation sign occurring in E to the smallest possible. By applying rules A1 - A5, it is possible to change E to an expression in which the scope of each negation sign (if any) in the new expression is only an atomic expression.

The second step is to standardize the variables in the expression. Since the variables are only dummy names and have no intuitive or formal semantic designation, we may replace any variable by any other variable. To standardize the variables, we simply change the variables in such a way that each quantifier governs a distinct variable that does not occur elsewhere outside the scope of that quantifier. For example,

$$(\exists x_1)((x_2)P_1^2 x_1 x_2 \vee (x_2)(P_1^2 x_1 x_2 \vee (\exists x_2)P_1^1 x_2))$$

would be rewritten as,

$$(\exists x_1)((x_2)P_1^2 x_1 x_2 \vee (x_3)(P_1^2 x_1 x_3 \vee (\exists x_4)P_1^1 x_4))$$

There will be, in general, a number of ways in which the variables may

be standardized because of the abundance of variables that may be used in making the replacements. But since variables serve as dummy names, all of these ways are essentially equivalent for our purposes.

The third step is to eliminate existential quantifiers.

However, before doing so, it will be desirable to reduce the scope of all quantifiers occurring in the expression to the smallest possible (not counting negation symbols). This is accomplished by repeated application of rules A6 - A9. For example, the expression $(x_1)(\exists x_2)(P_1^1 x_1 \vee \neg P_1^1 x_2)$ would become $(x_1)P_1^1 x_1 \vee (\exists x_2)\neg P_1^1 x_2$. Once the scope of the quantifiers is as small as possible, the next step is the elimination of all existential quantifiers. Suppose the sub-expression $(\exists x_i)E'$ occurs in the expression under consideration, where E' is some expression with at least one free occurrence of x_i . We must consider two cases in the removal of the quantifier. First, if this sub-expression does not occur within the scope of any universal quantifier, then we simply drop the quantifier " $(\exists x_i)$ " and replace each occurrence of x_i in E' by a constant symbol a_j which has not previously occurred anywhere in the expression. The second case arises when this sub-expression does occur in the scope of one or more universal quantifiers. Suppose the sub-expression occurs in the scope of universal quantifiers over the variables x_{j1}, \dots, x_{jn} . Then the existential quantifier is dropped and every free occurrence of x_i in E' is replaced by the term $f_k^n(x_{j1}, \dots, x_{jn})$, where f_k^n is a function symbol which has not previously occurred in the expression under

consideration. For example, the expression

$$(\exists x_1)(x_2)P_1^2 x_1 x_2 \vee (x_3)(\exists x_4)P_2^2 x_3 x_4$$

would become

$$(x_2)P_1^2 a_1 x_2 \vee (x_3)P_2^2 x_3 f_1^1(x_3)$$

The reason for first reducing the scope of each quantifier should now be obvious. If we remove the existential quantifier directly from an expression without reducing the scope of its quantifiers to a minimum, unnecessary function symbols may be introduced, or introduced functions may range over more variables than is necessary.

The fourth step is to place the expression in prenex normal form. This is done by applying rules A6 and A7 to move all of the universal quantifiers to the front of the entire expression. An expression in prenex normal form consists of a string of quantifiers (in our case all universal) called the prefix, followed by a quantifier free formula called a matrix. An easy method for uniquely specifying a prenex normal form of an expression consists of the following procedure. List all the quantifiers in the order of their occurrence in the expression from left to right. This list will be the prefix of the prenex normal form. The matrix is obtained by simply erasing all the quantifiers.

For the fifth step, the matrix of the resulting expression is placed in conjunctive normal form. An expression is in conjunctive normal form if it is of the form $E_1 \& \dots \& E_n$, where each E_i is of the form $E_{i1} \vee \dots \vee E_{im}$ and each E_{ij} is either an atomic expression or the negation of an atomic expression. An expression which is either atomic

or the negation of an atomic expression is called a literal. An expression which is a disjunction of literals is called a clause. An expression in conjunctive normal form is then a conjunction of clauses.

Repeated applications of rule All will place the matrix in conjunctive normal form. We may define a unique conjunctive normal form for expressions (the conjunctive normal form) by a variety of methods. We may impose an ordering on the expressions and take the conjunctive normal form to be the earliest in the ordering. Alternatively, we may specify a given order for applying the rules in obtaining the conjunctive normal form. This latter method will be used below.

For the sixth step, the universal quantifiers are simply dropped, leaving only the matrix in conjunctive normal form. This resulting expression is said to be the (u)qfcnf of the original expression.

We must now enquire into the logical relation between an expression E and its (u)qfcnf, E' . In obtaining E' , the only step at which the resultant was not LE to the expression with which the step began was step three. All other steps appealed only to the rules A1 through All and our conventions concerning variables and quantifiers. Thus, if it were not for that step, E would be LE to E' . However, results almost as strong can be proved. We will only state the results here, as proofs are somewhat complicated. (For more detailed discussion and proofs, see [15].) First, for any expression E_1 , if E_2 is obtained from E_1 by the removal of existential quantifiers in the manner indicated above, then E_1 is satisfied by any interpretation satisfying E_2 . Thus

in our case, for an expression E and its $(u)qfcnf$, E' , E is satisfied by any interpretation satisfying E' . However, the reverse is not the case. That is, for some examples we can specify an interpretation satisfying E but not satisfying E' . For example, let E be the expression $(x_1)(\exists x_2)P_1^2 x_1 x_2$. Then E' would be $P_1^2 x_1 f_1^1(x_1)$. We may now specify an interpretation I satisfying E but not E' . Let the domain of I be $\{1, 2\}$; let P_1^2 be true of $(1,2)$ and $(2,1)$, and no others; and let $f_1^1(1) = 1$ and $f_1^1(2) = 2$. Clearly I satisfies E but not E' . The only difficulty arises with the function symbols and constants which occur in E' but which do not occur in E . Thus we can make the following claim: Let E'' be any expression which does not contain any symbols occurring in E' but not occurring in E ; then every interpretation satisfying E satisfies E'' if and only if every interpretation satisfying E' satisfies E'' . Further, we can say that there is an interpretation satisfying E if and only if there is an interpretation satisfying E' .

These logical relations between an expression, E , and its $(u)qfcnf$, E' , will be used to justify the use of E' instead of E . As will be discussed below, we will be concerned with two cases. In the first case we will be trying to determine if an expression E is satisfiable; we will do this by examining E' instead of E . In the second case, we will be searching for expressions true in every interpretation in which E is true; since we will not be interested in the functions and constants introduced in E' , we can examine E' instead of E for this purpose and still be assured that our results hold for E .

We will now introduce the notion of uniform substitution of a term for a variable. Let x_i be any variable and t_1 and t_2 be any terms. Then the result of substituting t_2 in t_1 for x_i may be defined recursively as follows:

(S1) If t_1 is a constant, then the result of substituting t_2 in t_1 for x_i is just t_1 .

(S2) If t_1 is a variable, there are two cases. First, if $t_1 \neq x_i$, then the result of substituting t_2 in t_1 for x_i is just t_1 . Second, if $t_1 = x_i$, then the result is just t_2 .

(S3) Suppose $t_1 = f_j^k(t_{11}, \dots, t_{1k})$, and let t_{11}', \dots, t_{1k}' be the results of substituting t_2 in t_{11}, \dots, t_{1k} , respectively, for x_i . Then the result of substituting t_2 in t_1 for x_i is $f_j^k(t_{11}', \dots, t_{1k}')$.

Let E be any expression in which the variables have been standardized, let t be any term, and let x_i be any variable. Then the result of substituting t for x_i in E is defined as follows:

(S4) Let E be an atomic expression, say $P_j^{k_t} t_1 \dots t_k$, and let t_1', \dots, t_k' be the results of substituting t for x_i in t_1, \dots, t_k , respectively. Then $P_j^{k_t} t_1' \dots t_k'$ is the result of substituting t for x_i in E .

- (S5) Let E be of the form $E_1 \& E_2$ (or $E_1 \vee E_2$);
 let E_1' and E_2' be the results of substituting t for x_1 in E_1 and E_2 respectively.
 Then $E_1' \& E_2'$ (or $E_1' \vee E_2'$) is the result of substituting t for x_1 in E .
- (S6) Let E be of the form $\neg E_1$, and let E_1' be the result of substituting t for x_1 in E_1 . Then $\neg E_1'$ is the result of substituting t for x_1 in E .

Since we will be primarily concerned with substitutions in (u)qfcnf's, we will not extend the definition to cover quantifiers.

We will represent substitutions as sets of ordered pairs (t, x_i) , where x_i is the variable for which the substitution is being made and t is the term with which the variable is being replaced. There are few conventions concerning substitutions. Since any substitution of the form (x_i, x_i) will leave an expression unchanged, we may assume that such substitutions are not included in any given set of substitutions we may consider. Also, if a given set of substitutions contains two of the form (t, x_i) and (t', x_i) , where $t \neq t'$, then the substitution is ambiguous and hence improper. Thus we may assume that such pairs of substitutions are not included in any set we will consider. Henceforth, we will use the term "substitution" to refer to a set of substitutions (possibly empty). If σ is a (non-empty) substitution, then σ has the form $\{(t_1, x_{i_1}), \dots, (t_n, x_{i_n})\}$. The elements of σ are supposed to be

performed simultaneously.

Performing substitutions sequentially is represented by the concatenation operation. Let $\sigma = \{(t_{i_1}, x_{i_1}), \dots, (t_{i_n}, x_{i_n})\}$ and $\delta = \{(t_{j_1}, x_{j_1}), \dots, (t_{j_m}, x_{j_m})\}$ be two substitutions. Let $t_{i_1}', \dots, t_{i_n}'$ be the results of performing the substitution δ on each of the terms in σ . Let x_{j_p}, \dots, x_{j_r} be the variables for which substitution is being made in δ but not in σ . Then the concatenation of σ with δ , $\sigma\delta$, is the set $\{(t_{i_1}', x_{i_1}), \dots, (t_{i_n}', x_{i_n}), (t_{j_p}, x_{j_p}), \dots, (t_{j_r}, x_{j_r})\}$. The result of applying a substitution σ to an expression E is written $E\sigma$. It can be shown that for any two substitutions, σ and δ , $(E\sigma)\delta = E(\sigma\delta)$. It can also be shown that for any three substitutions σ , δ , and γ , that $(\sigma\delta)\gamma = \sigma(\delta\gamma)$. (See [5], pp. 55-56.)

Substitutions will be used for the purpose of "unifying" literals. Let $L = \{L_1, \dots, L_n\}$ be a set of literals. Then the set L is said to be unified by the substitution σ if

$$L_1\sigma = L_2\sigma = \dots = L_n\sigma$$

Unifying a set of literals then consists of finding a single substitution which when applied will make all of the literals identical. Necessary (but not sufficient) conditions for a set of literals to be unifiable are that all the literals must be constructed with the same predicate symbol, and they must all be atomic expressions or they must all be the negation of atomic expressions. Before attempting to find a substitution to unify a set of literals, these two conditions must be checked.

Suppose we are trying to find a substitution to unify a set of literals. For the purposes of the Resolution Principle, we do not want just any substitution -- we want the simplest one. We say that σ is the simplest substitution with some property if for every substitution δ with that property there is a substitution γ such that $\delta = \sigma\gamma$. For the purpose of finding the simplest substitution, we must introduce some ordering on the terms and expressions. The ordering is based on an ordering of the primitive symbols; we will place variables first, then constants, functions, predicates, and connectives in that order. The variables and constants are ordered according to subscripts, and the functions and predicates are ordered first according to superscript and then according to subscript. The order of the connectives is not important and some arbitrary order is assumed to be imposed. The ordering is extended to terms and expressions by putting terms before expressions; terms and expressions are then ordered by length, and two terms (or expressions) of the same length are ordered by their i th symbol, where the i th symbol is the first at which they differ.

It can be shown that if a set of literals is unifiable, then there is a simplest unifying substitution. (See [5], pp. 59-60.) Further, we can give an algorithm for obtaining this simplest substitution. Suppose we have two literals, L_1 and L_2 . If they are not the same, then there is some symbol position (reading from left to right) at which they disagree. That symbol position will be contained in a smallest subportion (expression or term) of L_1 and L_2 . L_1 and L_2 are said to first

disagree on that sub-portion. For examples, consider the following pairs:

- | | | |
|----------------------|-----------------------------|-------------------------|
| (1) $L_1: P_1^1 x_1$ | (2) $L_1: P_1^1 x_1$ | (3) $L_1: P_1^1 x_1$ |
| $L_2: P_2^1 x_1$ | $L_2: \neg P_1^1 x_1$ | $L_2: P_1^1 f_1^1(x_1)$ |
| (4) $L_1: P_1^1 x_1$ | (5) $L_1: P_1^2 x_1 x_2$ | |
| $L_2: P_1^1 x_2$ | $L_2: P_1^2 x_2 f_1^1(x_1)$ | |

In both (1) and (2), the two expressions first disagree on $\{L_1, L_2\}$. In (3) they first disagree on $\{x_1, f_1^1(x_1)\}$. In both (4) and (5), they disagree on $\{x_1, x_2\}$. For two literals, the set whose elements are the two sub-portions on which the literals first disagree is called the "disagreement set".

The disagreement set for a set of more than two literals is easily defined on this basis. Suppose we have a set $L = \{L_1, \dots, L_k\}$ of literals, where $k > 2$. Further suppose not all of the literals are identical. Let i be the least number such that there are two literals, L_m and L_n , that disagree at that symbol position. Let the disagreement set for L_m and L_n be $\{d_m, d_n\}$. For every other literal L_p in L , either L_p first disagrees with L_m on the sub-portion d_m , or L_p first disagrees with L_n on the sub-portion d_n , or both. The disagreement set of L is the union of all of these pair-wise disagreement sets. Intuitively, we find the first sub-portion at which there is disagreement between any pair of literals in L , and then we list the corresponding sub-portions of all of the members of L .

We may now give the algorithm for finding the simplest substitution σ_0 to unify a set L of literals, if there is such a

substitution. If there is no such substitution, the algorithm will tell us so:

- (1) Set $\sigma_1 = \emptyset$ and go to (2).
- (2) If the elements of $L\sigma_k$ are equal, set $\sigma_0 = \sigma_k$ and stop; otherwise go to (3).
- (3) Let s_k and t_k be the two earliest expressions in the lexical ordering of the disagreement set of $L\sigma_k$; if s_k is a variable and does not occur in t_k , set $\sigma_{k+1} = \sigma_k\{(t_k, s_k)\}$ and go to (2); otherwise stop. ([5], p. 59.)

It can be shown that if L can be unified, then the algorithm will stop in step (2) and give the simplest substitution; and if L cannot be unified, the algorithm will stop in step (3). (See [5], p. 59.)

The Resolution Principle is an inference rule which is applied to clauses. For the purposes of stating the rule, a clause will be identified with the set of literals composing it. Two literals are said to be complementary if one is the negation of the other. We may now define the notion of "resolvent" and state the rule.

A resolvent of two clauses C_1 and C_2 is a third clause D obtained as follows:

(i) If v_1, \dots, v_m are the variables of C_2 , and the highest variable of C_1 in the lexical order is u_k , let $\theta = \{(u_{k+1}, v_1), \dots, (u_{k+m}, v_m)\}$. (None of the variables in $C_2\theta$ occurs in C_1 .)

(ii) If there is a pair of sets of literals, $L = \{L_1, \dots, L_k\}$ and $M = \{M_1, \dots, M_n\}$ such that $L \subseteq C_1$ and $M \subseteq C_2$ and the set $\{L_1, \dots, L_k, \neg M_1\theta, \dots, \neg M_n\theta\}$ is unifiable, let σ_0 be the chosen simplest unifying substitution so that $L\sigma_0$ and $M\theta\sigma_0$ are [sets of] complementary literals; then D is the disjunction of the literals,

$$(C_1 - L)\sigma_0 \cup (C_2 - M)\theta\sigma_0$$

Resolution Principle. From any two clauses C_1 and C_2 infer a resolvent of C_1 and C_2 . ([5], pp. 56-57.)

We may note that any pair of clauses have only a finite number of resolvents, since there are only a finite number of pairs L and M of sets of literals and only one substitution for each pair. This leads to a convenient notation for resolvents. Let S be a set of clauses. Let $R(S)$ be the set of all clauses in S and all resolvents of pairs of clauses in S ; let $R^0(S) = S$, and let $R^{n+1}(S) = R(R^n(S))$, for $n \geq 0$. This notation allows us to keep track of the number of applications of the Resolution Principle required to obtain a particular resolvent.

There are two different ways in which the Resolution Principle may be employed, corresponding to the two different aspects of theorem proving discussed in section A of chapter I. First, we may be given a set of axioms A and some expression E and be asked to attempt to find a proof for E from the axioms; we may or may not know in advance that such a proof exists. In order to apply the Resolution Principle to this type of problem, we first form the conjunction of the axioms with the negation of E ; that is, letting CA stand for the conjunction of the axioms, we consider $E' = CA \ \& \ \neg E$. We then take the (u)qfcnf of E' , say E'' , and begin to apply the Resolution Principle to the clauses of E'' . In other words, we consider E'' as if it were a set of clauses instead of a conjunction of clauses, and we apply the Resolution Principle to this set. The expression E is a theorem just in case E' is contradictory. But E' is contradictory if and only if it is not satisfiable, and E' is not satisfiable if and only if E'' is not satisfiable. We know that E'' is not satisfiable if and only if there are contradictory clauses in E'' .

Intuitively, it seems that if this is the case, we should come up with two clauses, each consisting of a single literal, one being the negation of the other, so that their resolvent is \emptyset , the empty set. In fact, the following theorem can be proved (See [5], pp. 57-58.):

Theorem 13: Let S be a set of clauses. Then S is not satisfiable if and only if for some n , \emptyset is in $R^n(S)$.

This theorem expresses the soundness (truth preserving property) and completeness of the Resolution Principle when employed in the way described.

The second aspect of theorem proving to which the Resolution Principle can be applied is consequence generation. Again, suppose we are given a set of axioms A ; let CA be their conjunction, and let CA' be the (u)qfcnf of CA . We apply the Resolution Principle to the set of clauses obtained from CA' , and each resolvent is a consequence of CA' . That is, if E is any expression such that for some n , E is in $R^n(CA')$, then E is a consequence of CA' . The following theorem may be proved (See [12] , p. 181.):

Theorem 14: Let S be a set of clauses, and let E be a clause such that for some n , E is in $R^n(S)$. Then E is satisfied by every interpretation which satisfies every member of S .

This theorem expresses the soundness of the Resolution Principle when used as a consequence generator. However, the Resolution

Principle used as a consequence generator is complete only in a restricted sense. The following theorem may be proved (See [4], pp. 17-23.):

Theorem 15: Let S be a set of clauses and let E be a clause which is satisfied by every interpretation which satisfies all members of S . Then there is an n and a single clause C such that C is in $R^n(S)$, and E is satisfied by every interpretation which satisfies C .

To see that not every consequence of a set of axioms can be obtained directly by the Resolution Principle, simply note that the Resolution Principle does not provide for the introduction of new predicates not contained in the original clauses. Let E be any expression containing predicates not occurring in the axioms, and let A be one of the given axioms. Then $A \vee E$ is a consequence of the axiom set but would never be obtained by the Resolution Principle used as a consequence generator on the axiom set.

As we have already indicated in this section, not every consequence of the (u)qfcnf of E , call it E' , will be a consequence of E . However, it is assumed that for consequence generation we will keep track of new function symbols and constants introduced in E' during the removal of existential quantifiers and that all such functions and constants will be replaced by existential quantifiers in any consequence containing them. This will assure that the consequence is indeed a consequence of E .

B. Inductive Resolution

There are two methods by which the theory of f -rules may be applied to deductive resolution procedures to obtain inductive resolution procedures. (We will assume for now that we are only concerned with hypothesis generation.) The first method is indirect and was briefly discussed in section E of chapter II. Suppose we would like to generate hypotheses from which some set of expressions E all follow. First, we form the conjunction of the members of E , call it CE . Then we take the F transform of CE , say $F(CE)$. We then use $F(CE)$ in the consequence generator and take the F transform of the results. (Using the consequence generator first requires finding the $(u)qfcnf$ of $F(CE)$.) These expressions will be the hypotheses from which CE follows.

We must go into a bit more detail about taking the F transform of the results of the consequence generator. The results of the consequence generator will be $(u)qfcnf$, say C' . We actually want the F transform of an expression C whose $(u)qfcnf$ is C' . This can be obtained by applying the inverse of certain procedures used to obtain the $(u)qfcnf$ (see section A). First, we prefix C' by universal quantifiers over all free variables. We then distribute these quantifiers over $\&$ and \vee (but not over \neg) so that their scope is as small as possible (see rules A6 and A7 of the previous section). It is assumed that a list of constants and function symbols introduced in the removal of existential quantifiers has been kept; we will now replace all such constants and functions which remain (called "old" constants and functions henceforth)

by existential quantifiers. Each old constant symbol will be replaced by a new variable and an existential quantifier over that variable is prefixed to the front of the entire expression; the order of these existential quantifiers will not matter. We will now consider terms made of the old function symbols; they will be called old terms. If an old term contains no variables, it is treated like an old constant. If an old term contains variables, it will be within the scope of quantifiers over those variables; these quantifiers will be called covering quantifiers. For any two quantifiers having overlapping scope, the scope of one must fall entirely within the scope of the other. Thus there is an innermost covering quantifier, that is, one with smallest scope. The entire old term is replaced by a new variable and an existential quantifier over that variable is inserted just to the right of the innermost covering quantifier. The resulting expression may have vacuous quantifiers, and these can be dropped. By altering the order of the universal quantifiers before the introduction of the existential quantifiers, we may obtain different expressions upon reintroducing the existential quantifiers. However, we will assume that they are reintroduced in the order in which they were removed. Further, it may be the case that these "inverses" do not yield their originals upon application of the procedure of section A. However, it should be clear that any "inverse" is a deductive consequence of the original; further, the inverse is satisfiable if and only if the original is satisfiable, though

not necessarily by the same interpretation; finally, any consequence of one which does not contain the constants or functions removed, is a consequence of the other. In the following, we will refer to the expression obtained from C by this inverse procedure as the inverse-(u)qfcnf of C .

The indirect procedure suffers from a limitation inherent in Theorem 15. Since the Resolution Principle as a consequence generator is not complete in an unrestricted sense, then neither is the inductive procedure based on it in the indirect manner. The somewhat complicated completeness condition is stated below; its proof is immediate from Theorem 15 and the above comments.

Theorem 16: Let H and E be any expressions such that the (u)qfcnf of $F(H)$ is a clause and such that E is satisfied by every interpretation satisfying H . Let $F(E)'$ be the set of clauses obtained from (u)qfcnf of $F(E)$. Then for some n there is a single clause C in $R^n(F(E)')$ such that the F transform of the inverse-(u)qfcnf of C is satisfied by every interpretation satisfying H .

The more direct method of obtaining an inductive procedure from the Resolution Principle consists of stating an inductive Resolution Principle. In order to obtain such an inference rule, it will be necessary to introduce some more terminology. In particular, we must state how to obtain the (existential) quantifier free disjunctive normal form of an expression -- henceforth, the (e)qfdnf. A similar procedure to that described for converting to (u)qfcnf is employed in converting

to (e)qfdnf.

The first and second steps for obtaining (e)qfdnf are exactly the same as for obtaining (u)qfcnf.

The third step consists of removing universal quantifiers by a procedure analogous to that used in removing existential quantifiers before. Again we first reduce the scope of each quantifier to the smallest possible (not counting negation symbols). As before, there are two cases which must be considered. If the quantifier (x_i) does not occur with the scope of any existential quantifier, we simply drop the quantifier and replace all free occurrences of x_i in its scope by some new constant. On the other hand, if (x_i) occurs within the scope of existential quantifiers over the variables x_{j_1}, \dots, x_{j_n} , then we drop the (x_i) and replace each free occurrence of x_i by a term $f_p^n(x_{j_1}, \dots, x_{j_n})$, where f_p^n is a new function symbol.

The fourth step is to place the expression in prenex normal form, moving all of the existential quantifiers to the front of the quantifier free matrix.

For the fifth step, the matrix is placed in disjunctive normal form. We will call an expression consisting of a conjunction of literals an f -clause. The disjunctive normal form is then a disjunction of f -clauses.

For the sixth step, the existential quantifiers are dropped, leaving only the matrix. The result will be called the (e)qfdnf of the original.

We may define the f-resolvent of two f-clauses C_1 and C_2 to be a third f-clause D as follows:

- (i) If v_1, \dots, v_m are the variables of C_2 , and the highest variable in C_1 in the lexical order is u_k , let $\theta = \{(u_{k+1}, v_1), \dots, (u_{k+m}, v_m)\}$.
(None of the variables in $C_2\theta$ occurs in C_1 .)
- (ii) If there is a pair of sets of literals $L = \{L_1, \dots, L_k\}$ and $M = \{M_1, \dots, M_n\}$ such that $L \subseteq C_1$ and $M \subseteq C_2$ and the set $\{L_1, \dots, L_k, \neg M_1\theta, \dots, \neg M_n\theta\}$ is unifiable, let σ_0 be the chosen simplest unifying substitution so that $L\sigma_0$ and $M\theta\sigma_0$ are sets of complementary literals; then D is the conjunction of the literals.

$$(C_1 - L)\sigma_0 \cup (C_2 - M)\theta\sigma_0$$

Corresponding to the definition of the Resolution Principle, we may now define an inductive procedure which will be called the f-Resolution Principle.

f-Resolution Principle. From any two f-clauses C_1 and C_2 infer an f-resolvent of C_1 and C_2 .

We would now like to determine the relationship between the Resolution Principle and the f-Resolution Principle. In order to do so, we will first have to determine the relationship between the (u)qfcnf of E and the (e)qfdnf of E . The following theorems establish the desired

results. Since the proof of Theorem 17 is quite long, it is here omitted but may be found in the Appendix.

Theorem 17: Let E be any expression with no free variables. Then $F((u)qfcnf \text{ of } E) = (e)qfdnf \text{ of } F(E)$.

Theorem 18: Let E be any expression with no free variables. Then $(u)qfcnf \text{ of } F(E) = F((e)qfdnf \text{ of } E)$.

Proof: By Theorem 17, $F((u)qfcnf \text{ of } F(E)) = (e)qfdnf \text{ of } F(F(E))$. Taking the F transform of both sides of the equation gives us $F(F((u)qfcnf \text{ of } F(E))) = F((e)qfdnf \text{ of } F(F(E)))$. By Theorem 2, $(u)qfcnf \text{ of } F(E) = F((e)qfdnf \text{ of } E)$. Q.E.D.

Let E be any expression with no free variables. We will use $S(E)$ to denote the set of clauses obtained from E and $S'(E)$ to denote the set of f -clauses obtained from E . Let S' be any set of f -clauses. Then $fR(S')$ is the set of f -clauses in S' together with the f -resolvents of all pairs of clauses in S' . We define $fR^n(S')$ recursively as follows:

$$(i) \quad fR^0(S') = S'$$

$$(ii) \quad fR^n(S') = fR(fR^{n-1}(S')), \text{ for } n > 0.$$

The function fR^n is analogous to R^n defined above. The following theorems express the desired relationship between the Resolution Principle and the f -Resolution Principle.

Theorem 19: C_1 , C_2 , and D are clauses such that D is a resolvent of C_1 and C_2 if and only if $F(C_1)$, $F(C_2)$, and $F(D)$ are f -clauses

such that $F(D)$ is an f -resolvent of $F(C_1)$ and $F(C_2)$.

Proof: Since a clause is just a disjunction of literals, and since the F transform of a literal is the same literal, the F transform of a clause is just a conjunction of the literals in that clause. Hence, for any clause C , $F(C)$ is an f -clause. Similarly, for any f -clause H , $F(H)$ is a clause. Thus if $F(C)$ is an f -clause, then $F(F(C))$ is a clause; but by Theorem 2, $F(F(C)) = C$. Hence, C_1 , C_2 , and D are clauses if and only if $F(C_1)$, $F(C_2)$, and $F(D)$ are f -clauses. Since C_1 and $F(C_1)$ contain the same literals and C_2 and $F(C_2)$ contain the same literals, by the definitions of resolvent and f -resolvent, a pair of substitutions θ and

will yield a resolvent of C_1 and C_2 if and only if the same pair yield an f -resolvent of $F(C_1)$ and $F(C_2)$. Further, the resolvent obtained from C_1 and C_2 (if there is one) will contain the same literals as the f -resolvent obtained from $F(C_1)$ and $F(C_2)$ (if there is one). Thus, D is a resolvent of C_1 and C_2 if and only if $F(D)$ is an f -resolvent of $F(C_1)$ and $F(C_2)$. Q.E.D.

Theorem 20: For any expression E with no free variables, any clause C , and any natural number n , C is a member of $R^n(S(E))$ if and only if $F(C)$ is a member of $fR^n(S'(F(E)))$.

Proof: By induction on n . First note that C is a member of $S(E)$ if and only if C is a clause obtained from $(u)qfcnf$ of E . But since (by Theorem 17) $F((u)qfcnf \text{ of } E) = (e)qfdnf \text{ of } F(E)$, C is a clause obtained from $(u)qfcnf$ of E if and only if $F(C)$ is an f -clause obtained from

(e)qfdnf of $F(E)$. But the latter is true if and only if $F(C)$ is a member of $S'(F(E))$. For the basis step, let $n = 1$. C is a member of $R^1(S(E))$ if and only if C is a member of $S(E)$ or C is a resolvent of two members of $S(E)$. But we have just shown that C is a member of $S(E)$ if and only if $F(C)$ is a member of $S'(F(E))$. But then C is a resolvent of two members of $S(E)$ if and only if $F(C)$ is an f-resolvent of two members of $S'(F(E))$, by Theorem 19. Thus C is a member of $R^1(S(E))$ if and only if $F(C)$ is a member of $fR^1(S'(F(E)))$. For the induction hypothesis, assume the theorem is true for all n , $1 < n < m$. We must show the theorem holds for $n = m$. The only non-trivial case arises when C is a member of $R^m(S(E))$ but not of $R^{m-1}(S(E))$. By the induction hypothesis, C is not a member of $R^{m-1}(S(E))$ if and only if $F(C)$ is not a member of $fR^{m-1}(S'(F(E)))$. Now, for the case under consideration, C must be a resolvent of two clauses, C_1 and C_2 , both members of $R^{m-1}(S(E))$. By the induction hypothesis, C_1 and C_2 are both members of $R^{m-1}(S(E))$ if and only if $F(C_1)$ and $F(C_2)$ are both members of $fR^{m-1}(S'(F(E)))$. Further, by Theorem 19, C is a resolvent of C_1 and C_2 if and only if $F(C)$ is an f-resolvent of $F(C_1)$ and $F(C_2)$. Hence, C is a member of $R^m(S(E))$ if and only if $F(C)$ is a member of $fR^m(S'(F(E)))$. Q.E.D.

Theorem 21: For any expression E with no free variables, any f-clause C , and any natural number n , C is a member of $fR^n(S'(E))$ if and only if $F(C)$ is a member of $R^n(S(F(E)))$.

Proof: By Theorem 20, $F(C)$ is an element of $R^n(S(F(E)))$ if and

only if $F(F(C))$ is an element of $fR^n(S'(F(F(E))))$. But applying Theorem 2, the latter means that C is an element of $fR^n(S'(E))$. Q.E.D.

With these results relating the Resolution Principle and the f -Resolution Principle, we may obtain a completeness result for the f -Resolution Principle which is much neater than Theorem 16. The result is similar to Theorem 15, only it concerns f -resolvents. However, we must be careful about stating the semantic properties. The free variables in an f -clause must not be treated as if they were universally quantified. Rather, it must be remembered, they are the result of removing existential quantifiers. Hence, whenever we speak of the semantic properties of an f -clause, we must treat it as if it were preceded by existential quantifiers over the free variables in the f -clause. We will speak of this form of an f -clause, C , as the "existential quantification of C ".

Theorem 22: Let S' be a set of f -clauses and let E be an expression with no free variables which is falsified by every interpretation which falsifies the existential quantification of all members of S' . Then there is an n , $n \geq 0$, and a single f -clause C such that C is a member of $fR^n(S')$ and E is falsified by every interpretation which falsifies the existential quantification of C .

Proof: Let S' and E be given satisfying the hypothesis of the theorem. Let $D(S')$ be the expression formed by taking the disjunction of the existential quantification of the members of S' . Then $D(S') \vee \neg E$

must be LT. Thus $F(D(S') \vee \neg E)$ must be LF, by Theorem 5. Thus $F(D(S')) \& \neg F(E)$ is LF. Then every interpretation which satisfies $F(D(S'))$ must satisfy $F(E)$. The set of clauses obtained from $F(D(S'))$ is just $F(S')$, i.e., the set of the F transforms of the members of S' . Hence, by Theorem 15, there is an n and a single clause D such that D is an element of $R^n(F(S'))$ and such that $F(E)$ is satisfied by every interpretation satisfying D . Let D' be the universal quantification of D , obtained in a similar way as the existential quantification of an f -clause. Then $D' \& \neg F(E)$ is LF. By Theorem 5, $F(D' \& \neg F(E))$ is LT. That is, $F(D') \vee \neg F(F(E))$ is LT. Applying Theorem 2, we have $F(D') \vee \neg E$ is LT. This means that E must be falsified by every interpretation which falsifies $F(D')$. But $F(D')$ is just the existential quantification of $F(D)$. Further, by Theorem 20, $F(D)$ is a member of $fR^n(F(F(S')))$. Then by Theorem 2, $F(D)$ is a member of $fR^n(S')$. Q.E.D.

Thus, the f -Resolution Principle as an "inductive consequence" generator has a completeness property analogous to that of the Resolution Principle as a "deductive consequence" generator. The following theorem, analogous to Theorem 14, guarantees that the f -Resolution Principle truly is inductive, i.e., falsehood preserving. Again, we must be careful about free variables.

Theorem 23: Let S' be a set of f -clauses, and let E be an f -clause such that for some n , E is a member of $fR^n(S')$. Then the existential quantification of E is falsified by every interpretation which

falsifies the existential quantification of every member of S' .

Proof: Let E and S' be given satisfying the hypothesis of the theorem. Then by Theorem 21, $F(E)$ is a member of $R^n(F(S'))$. By Theorem 14, $F(E)$ is satisfied by every interpretation satisfying $F(S')$. Let $C(F(S'))$ be the conjunction of the universal quantifications of the members of $F(S')$, and let $F(E)'$ be the universal quantification of $F(E)$. Then $C(F(S')) \& \neg F(E)'$ is LF. By Theorem 5, $F(C(F(S')) \& \neg F(E)')$ is LT. That is, $F(C(F(S')) \vee \neg F(F(E)'))$ is LT. Thus $F(F(E)')$ is falsified by every interpretation which falsifies $F(C(F(S')))$. But $F(C(F(S')))$ is just the disjunction of the existential quantification of the members of S , while $F(F(E)')$ is just the existential quantification of E . Q.E.D.

Since resolution and f-resolution are exactly parallel, f-resolution may also be used in attempting to find proofs. Instead of trying to deduce a statement E from an axiom (or conjunction of axioms) A , we attempt to induce A from E . That is, we form the disjunction of E with the negation of A , i.e., $E \vee \neg A$. We then obtain (e)qfdnf of $E \vee \neg A$ and apply the f-Resolution Principle to the f-clauses obtained. If A really is an inductive consequence of E , then $E \vee \neg A$ will be LT, i.e., not falsifiable; then the existential quantification of (e)qfdnf of $E \vee \neg A$ will not be falsifiable, and for some n , \emptyset is a member of $fR^n(S'(E \vee \neg A))$. We may prove a result analogous to Theorem 13.

Theorem 24: Let S' be a set of f-clauses. Then the set whose members are the existential quantifications of the members of S' is not

falsifiable if and only if for some n , ϕ is a member of $fR^n(S')$.

Proof: As before, by forming the disjunction of the existential quantification of the members of S' we may show that the set whose members are the existential quantification of the members of S' is not falsifiable if and only if $F(S')$ is not satisfiable. $F(S')$ is not satisfiable if and only if for some n , ϕ is a member of $R^n(F(S'))$, by Theorem 13. But by Theorem 21, ϕ is an element of $R^n(F(S'))$ if and only if ϕ is an element of $fR^n(S')$. Q.E.D.

Thus we have developed an inductive resolution procedure which is parallel to deductive resolution. Inductive resolution may be used either to generate hypotheses (inductive consequences) or to provide "inductive" proofs, just as deductive resolution may be used either to generate deductive consequences or to provide deductive proofs. The soundness and completeness results for inductive resolution parallel those for deductive resolution.

Chapter IV: Power

A. The Parallel Proof Procedure

As we noted in chapter I, humans seldom solve the problem of deductively deriving a desired result from a given set of axioms by employing deductive rules alone. Usually some sort of dialectical process is employed, reasoning alternately deductively and inductively. In this section we will outline a method which formally incorporates this practice in a mechanical procedure.

The human problem solver attempts to construct a deduction of the desired result C from the given axiom (or conjunction of axioms) A by working from both ends towards some common middle ground. That is, he reasons "forward" from A and "backward" from C until the two chains of reasoning coincide at some point. Our procedure will parallel this process with one important difference. The human problem solver in reasoning backward from C employs essentially the same rules he employs in reasoning forward, only he applies them "in reverse", as it were. Thus by simply "inverting" his reasoning from C to, say, D, he obtains a deductive proof of C from D. This seems to be the sort of procedure Meltzer had in mind, as discussed in chapter II. In our procedure, the "backwards" reasoning from C will employ the f-rules corresponding to the deductive rules we use to reason "forward".

Suppose we are faced with the problem of determining whether or not C is a deductive consequence of A. Using the theory developed in

chapter II, there are four different routes we may take in our attempt.

We may try to construct any one of the four following proofs:

- (i) $A \vdash_T C$
- (ii) $F(A) \vdash_F F(C)$
- (iii) $C \vdash_F A$
- (iv) $F(C) \vdash_T F(A)$

Proofs corresponding to (i) and (ii) would differ only in notation and would in this sense be parallel; they would be of equal length, employing parallel rules at each step, and would thus be of equal difficulty. Similar remarks could be made concerning the proofs corresponding to (iii) and (iv). We thus have only two essentially distinct methods at our disposal, a deductive method (corresponding to (i)) and an inductive method (corresponding to (iii)). The fact that these two syntactical methods may be used to establish the desired result was based on semantical considerations (the Theorem of f-rules, chapter II) and was not dependent on any syntactical parallel between the two methods.

When faced with the problem of proving that C follows from A we could either deduce C from A, or we could induce A from C. For some systems of deduction there are examples of A and C for which it is shorter (in terms of required number of steps) to deduce C from A rather than induce A from C; for those same systems there will be other examples of A and C for which it is shorter to induce A from C rather than deduce C from A. But note that no matter which we do, if we are successful, we demonstrate that the truth of C is guaranteed by the truth of A (i.e.,

that C is true in every interpretation in which A is true). If we had some method of knowing in advance, for specified A and C , whether it would be shorter to use deduction or induction, our task would be a great deal easier. However, there is in general no way to tell in advance. The next best procedure then is to employ parallel processing.

The precise specification of the method is quite simple. Instead of constructing a single sequence of expressions according to one or the other of the sets of rules, we construct two sequences, one sequence beginning with A and produced according to the deductive rules and one sequence beginning with C and produced according to the inductive rules. The proof is successful if and only if there is a "convergence" of the two sequences. We will call such a proof a parallel proof. We must specify in more detail the meaning of "convergence".

For the human problem solver, convergence means obtaining the same expression, say D , on both lists. He is thus assured that $A \vdash_T D$ and $D \vdash_T C$, and hence that $A \vdash_T C$. In semantic terms, the truth of A guarantees the truth of D , the truth of D guarantees the truth of C , and hence the truth of A guarantees the truth of C . We could adopt the same notion of "convergence" for our parallel procedure. Suppose we obtain D on the deductive sequence beginning with A and also on the inductive sequence beginning with C . Then since the deductive rules are truth preserving, the truth of A would guarantee the truth of D . Since the inductive rules are falsehood preserving, the falsity of C would guarantee the falsity of D , and thus the truth of D would guarantee the truth of C .

Hence the truth of A would guarantee the truth of C. Employing this notion of convergence guarantees that there is a proof in our parallel procedure which is at least as short as the shortest of the purely deductive and purely inductive proofs. Since there is no intellectual effort involved in writing down the first element of a proof sequence, we will not count it as part of a proof when evaluating the length of the proof.

Let a complete set of deductive rules be given along with a set of inductive rules obtained by taking the F transform of the deductive rules. Let A and C be given such that the truth of A guarantees the truth of C. Let m be the number of steps (not counting the first step) in the shortest of the purely deductive proofs of C from A, and let n be the number of steps (not counting the first step) in the shortest of the purely inductive proofs of A from C. Then there is a parallel proof such that the sum of the steps in each sequence of the proof (not counting the first step in each) is less than or equal to $\min \{m, n\}$. To see that this is so, suppose m is the minimum of m and n. Then the shortest deductive proof of C from A is shorter than the shortest inductive proof of A from C. Let $(A, A_1, \dots, A_{m-1}, C)$ be the sequence constituting the shortest deductive proof. Then the pair of sequences given by $((A, A_1, \dots, A_{m-1}, C), (C))$ constitutes a parallel proof of the desired result. But the sum of the steps in the two sequences, not counting the first step in each, is just m. The case in which n is the minimum of m and n is similar. Thus there is a parallel proof at least as short as the shorter of the short-

est deductive proof and the shortest inductive proof.

The argument above is based on the fact that purely inductive proofs and purely deductive proofs may be regarded as just special cases of parallel proofs. However, the D common to both sequences in the parallel proof may be neither A nor C . In such cases, a shorter parallel proof than either the deductive version or the inductive version will result.

By slightly generalizing the notion of convergence, it may be possible to obtain even better results. In constructing the parallel sequences, we do not need to require for success that the same expression appear on both sequences. It is sufficient that an expression D appear on the deductive sequence and an expression D' appear on the inductive sequence such that the truth of D guarantees the truth of D' . For in this case, the truth of A guarantees the truth of D (from the fact that D is on the deductive sequence), the truth of D guarantees the truth of D' (our new convergence criterion) and the truth of D' guarantees the truth of C (from the fact that D' is on the inductive sequence). Hence, the truth of A guarantees the truth of C . If the same formula appears on both sequences, this is just a special case of the more generalized convergence condition.

The generalized convergence condition allows us to take advantage of circumstances in which one expression D' is easily recognizable as being a consequence of another expression D ; in such cases we will say that D' is an "easy consequence" of D . For example, if D' is either of

the form $D \vee E$ or of the form $E \vee D$, where E is any expression at all, then D' may be regarded as an easy consequence of D . Similarly, if D is either of the form $D' \& E$ or of the form $E \& D'$, where E is any formula at all, then D' may be regarded as an easy consequence of D . A precise specification of the generalized convergence criterion would require a listing of the syntactical structures D and D' such that D' is to be regarded as an easy consequence of D . We have the following general definition:

Definition: A parallel proof that C is a deductive consequence of A (or that A is an inductive consequence of C) consists of two sequences (A, A_1, \dots, A_n, D) and (C, C_1, \dots, C_m, D') such that the first sequence is a deductive proof of D from A , the second sequence is an inductive proof of D' from C , and such that D' is an easy consequence of D .

B. Parallel Proof and Resolution

We will now consider the application of the parallel proof procedure to the theory of resolution and f -resolution developed in chapter III. We will first consider resolution and f -resolution employed as deductive and inductive consequence generators.

As a beginning, we may attempt to deduce C from A , or alternatively, we may try to induce A from C . We will use $S(E)$ to denote the set of clauses obtained from the $(u)qfcnf$ of E and $S'(E)$ to denote the set of f -clauses obtained from the $(e)qfdnf$ of E . Then our deductive procedure attempts to find an n such that $S(C)$ is a subset of $R^n(S(A))$.

Alternatively, our inductive procedure attempts to find an m such that $S'(A)$ is a subset of $fR^m(S'(C))$.

In order to obtain the full power of the parallel proof procedure, we must employ slightly more general techniques. We must find a D (with no occurrences of constants and functions used in obtaining $(u)qfcnf$ of A) and a D' (with no occurrences of constants and functions used in obtaining $(e)qfdnf$ of C) such that (i) for some n , $S(D)$ is a subset of $R^n(S(A))$, (ii) for some m , $S'(D')$ is a subset of $fR^m(S'(C))$, and such that (iii) D' is an easy consequence of D . From (i) it follows that D is a deductive consequence of A , from (ii) it follows that C is a deductive consequence of D' , and thus from (iii) it follows that C is a deductive consequence of A .

There are several very real difficulties facing the implementation of these methods. First, we must establish patterns for "easy consequence" that are readily recognizable. But most important of all, we must be able to recognize such patterns when D is in $(u)qfcnf$ and D' is in $(e)qfdnf$. Our task then becomes at least doubly complicated with resolution.

The second way in which the Resolution Principle may be used is as a refutation procedure. In our attempt to show that C is a deductive consequence of A , we form the expression $A \ \& \ \neg C$, which we will refer to as E for short. We then try to find an n such that \emptyset is an element of $R^n(S(E))$. Alternatively, we may form $C \vee \neg A$, which we will refer to as E' for short. If C is a deductive consequence of A (true in every inter-

pretation in which A is true) then E' will not be falsifiable. Thus we seek for an m such that \emptyset is an element of $fR^m(S'(E'))$.

We must now consider the possibility of using the parallel proof procedure. For the languages we are considering, it is not difficult to show that for any expression P , if $\neg P$ is a deductive consequence of P , then $\neg P$ is LT, i.e., $\neg P$ is not falsifiable. Now, it is easy to show that $C \vee \neg A \text{ LE } \neg(A \& \neg C)$. Thus if we can show that $C \vee \neg A$ is a deductive consequence of $A \& \neg C$, then we are guaranteed that $C \vee \neg A$ is not falsifiable. But if $C \vee \neg A$ is not falsifiable, then C must be a deductive consequence of A . This chain of argument will form the basis of the parallel proof procedure for resolution used as a refutation procedure.

Again, let E stand for $A \& \neg C$ and E' stand for $C \vee \neg A$. Then for the parallel proof procedure we must find a D (with no occurrences of constants and functions used in obtaining $(u)qfcnf$ of E) and a D' (with no occurrences of constants and functions used in obtaining $(e)qfdnf$ of E') such that (i) for some n , $S(D)$ is a subset of $R^n(S(E))$, (ii) for some m , $S'(D')$ is a subset of $fR^m(S'(E'))$, and such that (iii) D' is an easy consequence of D . From (i) it follows that D is a deductive consequence of $A \& \neg C$, from (ii) it follows that $C \vee \neg A$ is a deductive consequence of D' , and thus from (iii) it follows that $C \vee \neg A$ is a deductive consequence of $A \& \neg C$. But this means that $C \vee \neg A$ is not falsifiable, and hence C is a deductive consequence of A .

Thus the parallel proof procedure may be applied to this latter

way of employing resolution and f-resolution. The difficulties discussed above concerning problems of recognizing convergence are unchanged in this application. Hence, before the parallel proof procedure can be effectively employed, these problems must be solved.

C. Problems for Further Research

There are various theoretical problems which remain to be investigated. One is concerned with Theorem 16. That theorem is a paradigm of unclarity and cumbersomeness. Is there a simpler, stronger, more straightforward completeness theorem that can be proved for that method?

A second theoretical problem is concerned with the extension of these methods to a broader class of languages. Is it possible to define f-resolution techniques for languages including identity, and for second and higher order predicate calculi?

A third theoretical problem is concerned with convergence criteria for the parallel proof procedure. A great deal of work needs to be done in the area of determining patterns for D and D' such that D' is an easy consequence of D and such that such patterns are readily recognizable when D is in $(u)qfcnf$ and D' is in $(e)qfdnf$. Only after this theoretical problem is solved will it be possible to apply the parallel proof procedure to resolution and f-resolution.

In addition to the theoretical problems, there are several interesting practical problems that remain to be solved. The first is

the actual implementation of the f-Resolution Principle. Such an implementation should be practically the same as, and present no more difficulty than, the implementation of the Resolution Principle.

The more interesting practical problems are concerned with the implementation of the parallel proof procedure. The crudest implementation is to run both resolution and f-resolution at the same time, in parallel. Each time a new resolvent is obtained, the set of f-resolvents would be scanned to check for convergence. Similarly, each time a new f-resolvent is obtained, the set of resolvents would be scanned to check for convergence.

But combining the two methods in the way indicated above will require that roughly one half the memory must be devoted to resolution and the other half must be devoted to f-resolution. This memory split places a severe limitation on the power of both parts of the parallel proof procedure. We suspect that much more powerful technique may be had by employing each routine separately. Again, suppose we want to show that C is a deductive consequence of A . We first apply resolution to $A \& \neg C$, looking for either \emptyset or $(u)qfcnf$ of $C \vee \neg A$. If unsuccessful before memory bounds are reached, we may then apply f-resolution to $C \vee \neg A$, looking for \emptyset or $(e)qfdnf$ of $A \& \neg C$. If unsuccessful we then step through the list of resolvents D_i obtained from $A \& \neg C$ and apply f-resolution to $C \vee \neg A \vee \neg D_i$, looking for \emptyset . If unsuccessful we step through the list of f-resolvents D_i' obtained from $C \vee \neg A$ and apply resolution to $A \& \neg C \& \neg D_i'$, looking for \emptyset . By continuing to use one

routine to generate goals for the other routine, we should work toward convergence if it exists. It may thus be possible to handle problems which would otherwise be beyond the scope of either routine separately.

Bibliography

- [1] Feigenbaum, E., B. Buchanan, and J. Lederberg, "Generality and Problem Solving : A Case Study Using the DENDRAL Program," Machine Intelligence 6, B. Meltzer and D. Michie, eds., American Elswier Publishing Company, Inc., New York, 1971, pp. 165-190.
- [2] Feigenbaum, E.A., and J. Feldman, eds., Computers and Thought, McGraw-Hill Book Company, New York, 1963.
- [3] Gold, E.M., "Limiting Recursion", The Journal of Symbolic Logic, vol. 30 (1965), pp. 28-48.
- [4] Lee, R.C., A Completeness Theorem and a Computer Program for Finding Theorems Derivable from Given Axioms, doctoral dissertation, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1967.
- [5] Luckham, D., "The Resolution Principle in Theorem-Proving," Machine Intelligence 1, N.L. Collins and D. Michie, eds., Oliver and Boyd, Edinburgh, 1967, pp. 47-61.
- [6] McCarthy, J., "Programms with Common Sense," Proceedings of the Symposium on Mechanisation of Thought Processes, ed. by H. von Foerster, Her Majesty's Stationery Office, London, 1959, pp. 75-84.

- [7] Meltzer, B., "The Semantics of Induction and the Possibility of Complete Systems of Inductive Inference," Artificial Intelligence, vol. 1 (1970), pp. 189-192.
- [8] Mendelson, E., Introduction to Mathematical Logic, D. Van Nostrand Company, Inc., New York, 1964.
- [9] Minsky, M., "Steps Toward Artificial Intelligence," Proceedings of the Institute of Radio Engineers, January, 1961, pp. 8-30.
- [10] Morgan, C.G., "Hypothesis Generation by Machine," Artificial Intelligence, vol. 2 (1971), pp. 179-187.
- [11] Newell, A., J.C. Shaw, and H. Simon, "Empirical Explorations with the Logic Theory Machine," Proceedings of the Western Joint Computer Conference, February, 1957, pp. 218-239.
- [12] Nilsson, N.J., Problem Solving Methods in Artificial Intelligence, McGraw-Hill Book Company, New York, 1971.
- [13] Robinson, J.A., "A Machine-Oriented Logic Based on the Resolution Principle," Journal of the Association for Computing Machinery, vol. 12 (1965), pp. 23-41.
- [14] Robinson, J.A., "An Overview of Mechanical Theorem Proving," Theoretical Approaches to Non-Numerical Problem Solving, ed. by R. Banerji and M. Mesarovic, Springer-Verlag New York, Inc.,

New York, 1970, pp. 2-20.

[15] Shoenfield, J.R., Mathematical Logic, Addison-Wesley Publishing Company, Don Mills, Ontario, 1967.

[16] Slagle, J.R., Artificial Intelligence : The Heuristic Programming Approach, McGraw-Hill Book Company, New York, 1971.

[17] Thomason, R.H., Symbolic Logic : An Introduction, Collier-Macmillan Limited, London, 1970.

[18] Whitehead, A.N., and B. Russell, Principia Mathematica, second edition, vol. I, Cambridge University Press, 1935.

Appendix

We will here present a proof of Theorem 17. The plan of the proof is to present a series of lemmas related to the steps used in obtaining (u)qfcnf and (e)qfdnf. We will begin the numbering of the lemmas with 1. We will assume as given the equivalences stated in chapter III.

Lemma 1: Let f_1 be defined recursively as follows:

- (i) $f_1(E) = E$, for any atomic expression E
- (ii) $f_1(E_1 \& E_2) = f_1(E_1) \& f_1(E_2)$
- (iii) $f_1(E_1 \vee E_2) = f_1(E_1) \vee f_1(E_2)$
- (iv) $f_1((x_1)E) = (x_1)f_1(E)$
- (v) $f_1((\exists x_1)E) = (\exists x_1)f_1(E)$
- (vi) $f_1(\neg E) = \neg E$, for any atomic expression E
- (vii) $f_1(\neg \neg E) = f_1(E)$
- (viii) $f_1(\neg (E_1 \& E_2)) = f_1(\neg E_1) \vee f_1(\neg E_2)$
- (ix) $f_1(\neg (E_1 \vee E_2)) = f_1(\neg E_1) \& f_1(\neg E_2)$
- (x) $f_1(\neg (x_1)E) = (\exists x_1)f_1(\neg E)$
- (xi) $f_1(\neg (\exists x_1)E) = (x_1)f_1(\neg E)$

Then for any expression E , $F(f_1(E)) = f_1(F(E))$.

Proof: By induction on the number of connectives and quantifiers in E . For the basis step, suppose E is atomic. Then by definition, $f_1(E) = F(E) = E$. Hence $F(f_1(E)) = f_1(F(E))$. For the induction step, suppose the theorem is true for any expression with fewer than n con-

nectives and quantifiers, for $n > 0$. Let E be any expression with n connectives and quantifiers. We will have five cases:

Case 1: $E = E_1 \& E_2$. Then

$$F(f_1(E)) = F(f_1(E_1) \& f_1(E_2)) = F(f_1(E_1)) \vee F(f_1(E_2))$$

Since E_1 and E_2 have fewer than n connectives and quantifiers, by the induction hypothesis we have $F(f_1(E_1)) = f_1(F(E_1))$ and $F(f_1(E_2)) = f_1(F(E_2))$. Thus,

$$\begin{aligned} F(f_1(E)) &= f_1(F(E_1)) \vee f_1(F(E_2)) = f_1(F(E_1) \vee F(E_2)) \\ &= f_1(F(E_1 \& E_2)) = f_1(F(E)) \end{aligned}$$

Case 2: $E = E_1 \vee E_2$. The proof is the same as that for case 1, interchanging "&" and " \vee " throughout.

Case 3: $E = (x_1)E_1$. Then

$$F(f_1(E)) = F((x_1)f_1(E_1)) = (\exists x_1)F(f_1(E_1))$$

Since E_1 has fewer than n connectives and quantifiers, by the induction hypothesis we have $F(f_1(E_1)) = f_1(F(E_1))$. Thus,

$$\begin{aligned} F(f_1(E)) &= (\exists x_1)f_1(F(E_1)) = f_1((\exists x_1)F(E_1)) = f_1(F((x_1)E_1)) \\ &= f_1(F(E)) \end{aligned}$$

Case 4: $E = (\exists x_1)E_1$. The proof is the same as that for case 3, interchanging " (x_1) " and " $(\exists x_1)$ " throughout.

Case 5: $E = \neg E_1$. We will have six subcases, depending on the complexity of E_1 .

subcase 5.1: E_1 is atomic. Then

$$F(f_1(E)) = F(\neg E_1) = \neg F(E_1) = \neg E_1 = E$$

Further,

$$f_1(F(E)) = f_1(\neg F(E_1)) = f_1(\neg E_1) = \neg E_1 = E$$

Hence, $F(f_1(E)) = f_1(F(E))$.

subcase 5.2: $E_1 = \neg E_2$. Then $F(f_1(E)) = F(f_1(E_1))$.

Since E_1 has fewer than n connectives, by the induction hypothesis we have $F(f_1(E_1)) = f_1(F(E_1))$. Thus $F(f_1(E)) = f_1(F(E_1))$. Further, $f_1(F(E)) = f_1(\neg \neg F(E_1)) = f_1(F(E_1))$. Hence $F(f_1(E)) = f_1(F(E))$.

subcase 5.3: $E_1 = E_2 \& E_3$. Then

$$F(f_1(E)) = F(f_1(\neg E_2) \vee f_1(\neg E_3)) = F(f_1(\neg E_2)) \& F(f_1(\neg E_3))$$

Since $\neg E_2$ and $\neg E_3$ have fewer than n connectives, by the induction hypothesis we have, $F(f_1(\neg E_2)) = f_1(F(\neg E_2))$ and $F(f_1(\neg E_3)) = f_1(F(\neg E_3))$. Thus,

$$\begin{aligned} F(f_1(E)) &= f_1(F(\neg E_2)) \& f_1(F(\neg E_3)) = f_1(\neg F(E_2)) \& f_1(\neg F(E_3)) \\ &= f_1(\neg (F(E_2) \vee F(E_3))) = f_1(\neg F(E_2 \& E_3)) \\ &= f_1(F(\neg (E_2 \& E_3))) = f_1(F(E)) \end{aligned}$$

subcase 5.4: $E_1 = E_2 \vee E_3$. The proof is the same as that for subcase 5.3, interchanging "&" and " \vee " throughout.

subcase 5.5: $E_1 = (x_i)E_2$. Then

$$F(f_1(E)) = F((\exists x_i)f_1(\neg E_2)) = (x_i)F(f_1(\neg E_2))$$

Since $\neg E_2$ has fewer than n connectives, by the induction hypothesis we have $F(f_1(\neg E_2)) = f_1(F(\neg E_2))$. Thus,

$$\begin{aligned} F(f_1(E)) &= (x_i)f_1(F(\neg E_2)) = (x_i)f_1(\neg F(E_2)) = f_1(\neg (\exists x_i)F(E_2)) \\ &= f_1(\neg F((x_i)E_2)) = f_1(F(\neg (x_i)E_2)) = f_1(F(E)) \end{aligned}$$

subcase 5.6: $E_1 = (\exists x_i)E_2$. The proof is the same as that for subcase 5.5, interchanging " (x_i) " and " $(\exists x_i)$ " throughout. Q.E.D.

Lemma 2: Let f_2 be the function which standardizes the variables in an expression. That is, let E be a given expression with m quantifiers, and let $X = \{x_1', \dots, x_m'\}$ be a set of variables none of which occur in E . Then f_2 is defined recursively as follows:

- (i) $f_2(E_1) = E_1$, for any atomic expression E_1
- (ii) $f_2(\neg E_1) = \neg f_2(E_1)$
- (iii) $f_2(E_1 \& E_2) = f_2(E_1) \& f_2(E_2)$
- (iv) $f_2(E_1 \vee E_2) = f_2(E_1) \vee f_2(E_2)$
- (v) $f_2((x_i)E_1) = (x_j')f_2(E_1(x_i/x_j'))$, where x_j' is an element of X that has not been used before and $E_1(x_i/x_j')$ is the result of uniformly replacing all free occurrences of x_i in E_1 by x_j' .
- (vi) $f_2((\exists x_i)E_1) = (\exists x_j')f_2(E_1(x_i/x_j'))$, where x_j' and $E_1(x_i/x_j')$ are characterized as in (v).

Then $F(f_2(E)) = f_2(F(E))$.

Proof: By induction on the number of connectives and quantifiers in E . For the basis step, suppose E is atomic. Then $F(E) = f_2(E) = E$. Hence $F(f_2(E)) = f_2(F(E))$. For the induction step, assume the theorem holds for all expressions with fewer than n connectives and quantifiers, for $n > 0$. Suppose E has n connectives and quantifiers. We will have five cases:

Case 1: $E = \neg E_1$. Then

$$F(f_2(E)) = F(\neg f_2(E_1)) = \neg F(f_2(E_1))$$

Since E_1 has fewer than n connectives, we have by the induction hypothesis $F(f_2(E_1)) = f_2(F(E_1))$. Thus,

$$F(f_2(E)) = \neg f_2(F(E_1)) = f_2(\neg F(E_1)) = f_2(F(\neg E_1)) = f_2(F(E))$$

Case 2: $E = E_1 \& E_2$. Then

$$F(f_2(E)) = F(f_2(E_1) \& f_2(E_2)) = F(f_2(E_1)) \vee F(f_2(E_2))$$

Since E_1 and E_2 have fewer than n connectives and quantifiers, we have by the induction hypothesis $F(f_2(E_1)) = f_2(F(E_1))$ and $F(f_2(E_2)) = f_2(F(E_2))$. Thus,

$$\begin{aligned} F(f_2(E)) &= f_2(F(E_1)) \vee f_2(F(E_2)) = f_2(F(E_1) \vee F(E_2)) \\ &= f_2(F(E_1 \& E_2)) = f_2(F(E)) \end{aligned}$$

Case 3: $E = E_1 \vee E_2$. The proof is the same as that for case 2, interchanging "&" and " \vee " throughout.

Case 4: $E = (x_i)E_1$. Then

$$F(f_2(E)) = F((x_j')f_2(E_1(x_i/x_j')))) = (\exists x_j')F(f_2(E_1(x_i/x_j'))))$$

Since $E_1(x_i/x_j')$ has fewer than n connectives and quantifiers, by the induction hypothesis we have $F(f_2(E_1(x_i/x_j')))) = f_2(F(E_1(x_i/x_j'))))$.

Thus,

$$\begin{aligned} F(f_2(E)) &= (\exists x_j')f_2(F(E_1(x_i/x_j')))) = f_2((\exists x_i)F(E_1)) \\ &= f_2(F((x_i)E_1)) = f_2(F(E)) \end{aligned}$$

Case 5: $E = (\exists x_i)E_1$. The proof is the same as that for case 4, interchanging " $(\forall x_i)$ " and " (x_i) " and interchanging " $(\forall x_j')$ " and " (x_j') " throughout. Q.E.D.

Lemma 3.1: Let $f_{3.1}$ be the function that distributes quantifiers from larger scope to smaller scope, where possible, according to rules A6 - A9. Then $F(f_{3.1}(E)) = f_{3.1}(F(E))$.

Proof: Let E, E^1, E^2, \dots, E^n be a sequence of formulas obtained from E by successive applications of rules A6 - A9, such that $f_{3.1}(E) = E^n$. We will show that there is a sequence of applications of rules A6 - A9 beginning with $F(E)$ and terminating in $F(E^n)$ such that $f_{3.1}(F(E)) = F(E^n)$. This will guarantee that $F(f_{3.1}(E)) = F(E^n) = f_{3.1}(F(E))$. Consider the sequence $F(E), F(E^1), \dots, F(E^n)$ obtained by taking the F transform of each member of the original sequence. It is a trivial matter to verify that if rule A6 were used to go from E^i to E^{i+1} then rule A9 will go from $F(E^i)$ to $F(E^{i+1})$; if rule A7 were used in the original, then rule A8 may be used in the new; if rule A8 were used in the original, then rule A7 may be used in the new; and finally, if rule A9 were used in the original, then rule A6 may be used in the new. Since $E^n = f_{3.1}(E)$, no more rules are applicable to E^n to reduce the scope of the quantifiers there. But this means that no more rules are applicable to $F(E^n)$, and hence $F(E^n) = f_{3.1}(F(E))$. Q.E.D.

Lemma 3.2: Let $f_{3.2}$ be the function which eliminates existential quantifiers in the manner specified by step 3 for obtaining (u)qfcnf. Let $f_{3.2}'$ be the function which eliminates universal quantifiers in the manner specified by step 3 for obtaining (e)qfdnf. Then if the same function and constant symbols are used for corresponding quantifiers, $F(f_{3.2}(E)) = f_{3.2}'(F(E))$.

Proof: If E has no existential quantifiers, then $f_{3.2}(E) = E$. Thus $F(f_{3.2}(E)) = F(E)$. But if E has no existential quantifiers then $F(E)$ has no universal quantifiers, and hence $f_{3.2}(F(E)) = F(E)$. Thus in this case $F(f_{3.2}(E)) = f_{3.2}(F(E))$. On the other hand, suppose E has existential quantifiers occurring in it. Consider an arbitrary sub-expression, E' , of E beginning with an existential, say $(\exists x_1)E_1$. Then this subformula would be replaced by $E_1(x_1/t)$, where t is the appropriate term. The F transform would then have the sub-expression $F(E_1(x_1/t))$. Now, $F(E)$ would have, corresponding to E' , the sub-expression $F(E')$, i.e., $(x_1)F(E_1)$. Applying $f_{3.2}'$ to $F(E)$ would require the removal of the universal quantifier. Using the same term, we would obtain $F(E_1(x_1/t))$. Hence $F(f_{3.2}(E)) = f_{3.2}'(F(E))$. Q.E.D.

Lemma 3: Let $f_3(E) = f_{3.2}(f_{3.1}(E))$, and $f_3'(E) = f_{3.2}'(f_{3.1}(E))$. Then $F(f_3(E)) = f_3'(F(E))$.

Proof: Using Lemmas 3.1 and 3.2, we have,

$$\begin{aligned} F(f_3(E)) &= F(f_{3.2}(f_{3.1}(E))) = f_{3.2}'(F(f_{3.1}(E))) \\ &= f_{3.2}'(f_{3.1}(F(E))) = f_3'(F(E)) \end{aligned}$$

Q.E.D.

Lemma 4: Let f_4 be the function which moves all quantifiers to the front of an expression in the order of their occurrence in that expression. Then $F(f_4(E)) = f_4(F(E))$.

Proof: If there are no quantifiers in E , then there are none in $F(E)$. Hence $f_4(E) = E$ and $f_4(F(E)) = F(E)$. Thus $F(f_4(E)) = F(E) = f_4(F(E))$. On the other hand, suppose E does contain quantifiers. Let

the prefix of $f_4(E)$ be $(Q_1x_{i_1})...(Q_nx_{i_n})$ and the matrix be E' . Then $F(f_4(E)) = (Q_1'x_{i_1})...(Q_n'x_{i_n})F(E')$, where $(Q_j'x_{i_j})$ is existential if $(Q_jx_{i_j})$ was universal and universal otherwise. In $F(E)$, the quantifiers will occur in the same order as in E , but existential will be changed to universal and universal to existential. Thus the prefix for $f_4(F(E))$ will be the same as that for $F(f_4(E))$. Further, the result of erasing quantifiers from $F(E)$ will simply yield $F(E')$. Hence, $F(f_4(E)) = f_4(F(E))$. Q.E.D.

Lemma 5.1: Let the functions $f_{5.1}$ and $f_{5.1}'$ be defined recursively as follows:

- (i) $f_{5.1}(E) = E$, for any atomic expression E
- (ii) $f_{5.1}(\neg E) = \neg E$
- (iii) $f_{5.1}(E_1 \& E_2) = f_{5.1}(E_1) \& f_{5.1}(E_2)$
- (iv) $f_{5.1}(E_1 \vee E_2) = f_{5.1}(E_1) \vee E_2$, for E_2 atomic
- (v) $f_{5.1}(E_1 \vee \neg E_2) = f_{5.1}(E_1) \vee \neg E_2$
- (vi) $f_{5.1}(E_1 \vee (E_2 \vee E_3)) = f_{5.1}(E_1) \vee f_{5.1}(E_2 \vee E_3)$
- (vii) $f_{5.1}(E_1 \vee (E_2 \& E_3)) = f_{5.1}(E_1 \vee E_2) \& f_{5.1}(E_1 \vee E_3)$
- (viii) $f_{5.1}((Qx_i)E) = (Qx_i)f_{5.1}(E)$, for any quantifier over any variable x_i
- (i') $f_{5.1}'(E) = E$, for any atomic expression E
- (ii') $f_{5.1}'(\neg E) = \neg E$
- (iii') $f_{5.1}'(E_1 \vee E_2) = f_{5.1}'(E_1) \vee f_{5.1}'(E_2)$
- (iv') $f_{5.1}'(E_1 \& E_2) = f_{5.1}'(E_1) \& E_2$, for E_2 atomic
- (v') $f_{5.1}'(E_1 \& \neg E_2) = f_{5.1}'(E_1) \& \neg E_2$

- (vi') $f_{5.1}'(E_1 \& (E_2 \& E_3)) = f_{5.1}'(E_1) \& f_{5.1}'(E_2 \& E_3)$
 (vii') $f_{5.1}'(E_1 \& (E_2 \vee E_3)) = f_{5.1}'(E_1 \& E_2) \vee f_{5.1}'(E_1 \& E_3)$
 (viii') $f_{5.1}'((Qx_i)E) = (Qx_i)f_{5.1}'(E)$, for any quantifier
 over any variable x_i

Let E be any expression in prenex normal form such that the scope of each negation symbol occurring in E (if any) is at most an atomic expression. Then $F(f_{5.1}(E)) = f_{5.1}'(F(E))$.

Proof: Simply note that for each item in the definition of the two functions, the definition of $f_{5.1}'$ is just the F transform of the definition of $f_{5.1}$. The proof by induction on the connectives and quantifiers in E is then trivial. Q.E.D.

Lemma 5.2: Let the functions $f_{5.2}$ and $f_{5.2}'$ be defined recursively as follows:

- (i) $f_{5.2}(E) = E$, for any atomic expression E
 (ii) $f_{5.2}(\neg E) = \neg E$
 (iii) $f_{5.2}(E_1 \& E_2) = f_{5.2}(E_1) \& f_{5.2}(E_2)$
 (iv) $f_{5.2}(E_1 \vee E_2) = E_1 \vee f_{5.2}(E_2)$, for E_1 atomic
 (v) $f_{5.2}(\neg E_1 \vee E_2) = \neg E_1 \vee f_{5.2}(E_2)$
 (vi) $f_{5.2}((E_1 \vee E_2) \vee E_3) = f_{5.2}(E_1 \vee E_2) \vee f_{5.2}(E_3)$
 (vii) $f_{5.2}((E_1 \& E_2) \vee E_3) = f_{5.2}(E_1 \vee E_3) \& f_{5.2}(E_2 \vee E_3)$
 (viii) $f_{5.2}((Qx_i)E) = (Qx_i)f_{5.2}(E)$, for any quantifier
 over any variable x_i
 (i') $f_{5.2}'(E) = E$, for any atomic expression E
 (ii') $f_{5.2}'(\neg E) = \neg E$

- (iii') $f_{5.2}'(E_1 \vee E_2) = f_{5.2}'(E_1) \vee f_{5.2}'(E_2)$
- (iv') $f_{5.2}'(E_1 \& E_2) = E_1 \& f_{5.2}'(E_2)$, for E_1 atomic
- (v') $f_{5.2}'(\neg E_1 \& E_2) = \neg E_1 \& f_{5.2}'(E_2)$
- (vi') $f_{5.2}'((E_1 \& E_2) \& E_3) = f_{5.2}'(E_1 \& E_2) \& f_{5.2}'(E_3)$
- (vii') $f_{5.2}'((E_1 \vee E_2) \& E_3) = f_{5.2}'(E_1 \& E_3) \vee f_{5.2}'(E_2 \& E_3)$
- (viii') $f_{5.2}'((Qx_i)E) = (Qx_i)f_{5.2}'(E)$ for any quantifier
over any variable x_i

Let E be any expression in prenex normal form such that the scope of each negation symbol occurring in E (if any) is at most an atomic expression. Then $F(f_{5.2}(E)) = f_{5.2}'(F(E))$.

Proof: Simply note that for each item in the definition of the two functions, the definition of $f_{5.2}'$ is just the F transform of the definition of $f_{5.2}$. The proof by induction on the connectives and quantifiers in E is then trivial. Q.E.D.

Lemma 5: Let $f_5(E)$ be defined as $f_{5.2}(f_{5.1}(E))$, and let $f_5'(E)$ be defined as $f_{5.2}'(f_{5.1}'(E))$. Let E be any expression in prenex normal form such that the scope of each negation symbol occurring in E (if any) is at most an atomic expression. Then $F(f_5(E)) = f_5'(F(E))$.

Proof: This result is an immediate consequence of Lemmas 5.1 and 5.2. Q.E.D.

Lemma 5.3: Let $f_5^{(n)}$ and $f_5'^{(n)}$ be defined recursively as follows:

$$(i) \quad f_5^{(1)}(E) = f_5(E)$$

$$(ii) \quad f_5^{(n+1)}(E) = f_5(f_5^{(n)}(E))$$

$$(i') \quad f_5^{(1)}(E) = f_5'(E)$$

$$(ii') \quad f_5^{(n+1)}(E) = f_5'^{(n)}(E)$$

Let E be any expression in prenex normal form such that the scope of each negation symbol occurring in E (if any) is at most an atomic expression. Then $f_5^{(n)}(E) = f_5^{(n-1)}(E)$ if and only if $f_5'^{(n)}(F(E)) = f_5'^{(n-1)}(F(E))$, for any $n > 1$.

Proof: Since the function F is one-to-one and onto, $f_5^{(n)}(E) = f_5^{(n-1)}(E)$ if and only if $F(f_5^{(n)}(E)) = F(f_5^{(n-1)}(E))$. But by repeated application of Lemma 5, we have $F(f_5^{(n)}(E)) = f_5'^{(n)}(F(E))$ and $F(f_5^{(n-1)}(E)) = f_5'^{(n-1)}(F(E))$. Q.E.D.

Lemma 6: Let f_6 be defined recursively as follows:

$$(i) \quad f_6(E) = E, \text{ for any atomic expression } E$$

$$(ii) \quad f_6(\neg E) = \neg f_6(E)$$

$$(iii) \quad f_6(E_1 \vee E_2) = f_6(E_1) \vee f_6(E_2)$$

$$(iv) \quad f_6(E_1 \& E_2) = f_6(E_1) \& f_6(E_2)$$

$$(v) \quad f_6((Qx_i)E) = f_6(E), \text{ for any quantifier over any variable } x_i$$

Then for any expression E , $F(f_6(E)) = f_6(F(E))$.

Proof: Note that since f_6 indiscriminately erases quantifiers, it makes no difference whether they have been changed by F or not. The proof is then a trivial induction over the connectives and quantifiers in E . Q.E.D.

Theorem 17: Let E be any expression with no free variables.

Then $F((u)qfcnf \text{ of } E) = (e)qfdnf \text{ of } F(E)$.

Proof: Let n be such that $f_5^{(n-1)}(f_4(f_3(f_2(f_1(E)))))) = f_5^{(n)}(f_4(f_3(f_2(f_1(E))))))$. Then $(u)qfcnf$ of E is given by $f_6(f_5^{(n)}(f_4(f_3(f_2(f_1(E))))))$. Further, $(e)qfdnf$ of $F(E)$ is given by $f_6(f_5^{(n)}(f_4(f_3(f_2(f_1(F(E))))))$. (By Lemma 5.3, the " n " is the same in each case.) By Lemmas 1, 2, 3, 4, 5, and 6, $F((u)qfcnf \text{ of } E) = (e)qfdnf \text{ of } F(E)$. Q.E.D.

B30015